信息安全原理 HW1:Password Cracking

密文 1：凯撒(Caesar)密码

Ciphertext ： FBUQIUUDSHOFJOEKHDQCUMYJXJXUIQCUAUOQDTKFBEQTJEBUQHDYDWYDPZK

解密方式：对以上的密文字符串进行 26 次的循环（25 种位移的可能，第 26 次是密文本身），在循环中对每个字符加一并得到新的一种字符串（位移一次）打印出来，然后再对字符串进行识别，最终得出的结果在第 10 行，即向左位移了 10 次或可说是向右位移 16 次。

明文：PLEASE ENCRYPT YOUR NAME WITH THE SAME KEY AND UPLOAD TO LEARNING IN ZJU

```
shift = 10 明文：PLEASEENCRYPTYOURNAMEWITHTHESAMEKEYANDUPLOADTOLEARNINGINZJU
```

代码：

```
#include<iostream>
#include<cstring>
using namespace std;

int main()
{
	char str[100] = "FBUQIUUDSHOFJOEKHDQCUMYJXJXUIQCUAUOQDTKFBEQTJEBUQHDYDWYDPZK";
	cout << "密文：" << str << endl;
	int count_shift = 0;
	for(int i = 0; i <= 25 ; i++){
		//test 26 type of possible , shift 1 / 2 / 3 / 4 / ...
		for(int j = 0; j < strlen(str) ; j++){
			if(str[j] == 'Z') {
			   //如果遇到字母 Z ，则换成字母 A ，已经加一了所以跳到下一个循环
				str[j] = str[j] - 'Z' + 'A';
				continue;
			}
			str[j] = str[j] + 1;
		}
		count_shift++;
		cout << "shift = " << count_shift << " 明文：" << str << endl << endl;
	}
}
```

加密我的名字，明文:YAPXIYUAN，密文: OQFNYOKQD

```
我的名字的密文：OQFNYOKQD
```

```
//ENCRYPT MY NAME
	char name[20] = "YAPXIYUAN";
	for(int k = 0; k < strlen(name) ; k++){
		//1.A 2.B 3.C 4.D ...
		//如果位移后大于 26 则除以 26 得余数再加'A'
		name[k] = ( name[k] - 'A' + 16 ) % 26 + 'A' ;
	}
	cout << "我的名字的密文：" << name << endl;
```

密码 2：维吉尼亚(Vignere)密码

Ciphertext:ktbueluegvitnthuexmonveggmrcgxptlyhhjaogchoemqchpdnetxupbqntietiabpsmaoncnwvoutiugtagmmqsxtvxaoniiogtagmbpsmtuvvihpstpdvcrxhokvhxotawswquunewcgxptlcrxtevtubvewcnwwsxfsnptswtagakvoyyak

解密方式：

第一步：寻找密钥的长度，在加密的字符串中查找重复的字符串并记录该字符串的长度，长度取最大公约数(GCD)。

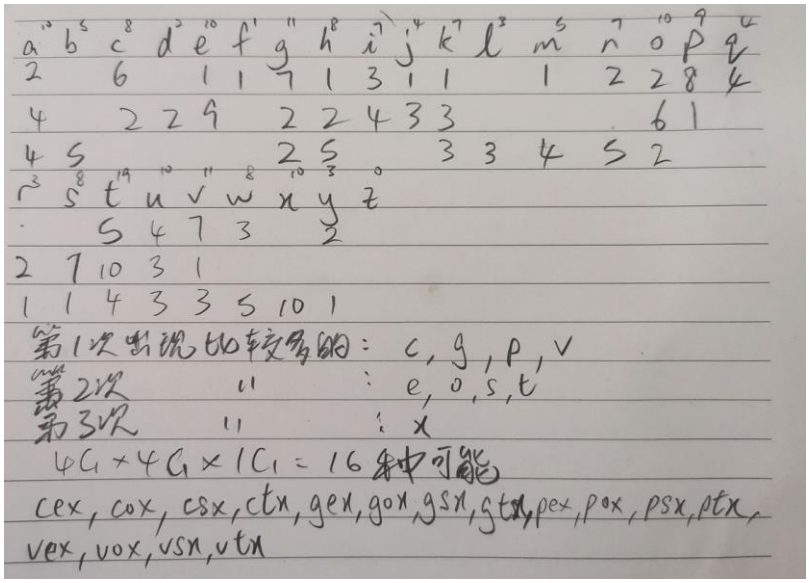
代码：

```
#include<iostream>
#include<vector>
#include<string>
#include<iterator>
using namespace std;

string Find_Repeat_String(string str)
{
	string temp,str1;
	int maxlen = 0;
	for(int i = 0 ; i < str.length() ; i++){
		for(int j = str.length() - i ; j != 0 ; j--){
			temp = str.substr(i,j); // get the sub str
ing from last index to i
			int pos1 = str.find(temp);
			int pos2 = str.rfind(temp); // see whether
 got repeat or not and get the position
			int length = temp.length();
			if(pos1 != pos2 && length > maxlen){ //if
 the two position not same and length is maximum
				str1 = temp;
				maxlen = length;
			}
			//else continue;
		}
	}
	return str1;
}
int main(){
	string str = "ktbueluegvitnthuexmonveggmrcgxptlyhhjaogchoemqch
pdnetxupbqntietiabpsmaoncnwvoutiugtagmmqsxtvxaoniiogtagmbpsmtuvvihpstpdvc
rxhokvhxotawswquunewcgxptlcrxtevtubvewcnwwsxfsnptswtagakvoyyak";
	string temp = str; //string that let to find
```

```
        string  temp1;  //find  string
        vector<string>  vs;
        //find  repeat  string  and  length
        temp1  =  Find_Repeat_String(temp);
        while(temp1.length()  >  2){  //the  minimum  length  is  3
                cout  <<  temp1  <<  "  ";
                vs.push_back(temp1);
                vector<int>  v;
                string  temp2  =  str;  //  full  string
                int  pos1  =  temp2.find(temp1);
                while(pos1  !=  string::npos){
                        v.push_back(pos1  +  temp1.length());//push  the
  position  to  vector
                        temp2.erase(0,  pos1+temp1.length());  //erase  t
he  string  which  before  the  position
                        pos1  =  temp2.find(temp1);//find  next  one
                }
                vector<int>::iterator  ite;
                for(ite  =  v.begin()  ;  ite  <  v.end()  ;  ite++){
                        cout  <<  *ite  <<  "  ";
                }
                //delete  the  repeated  string
                int  pos2  =  temp.find(temp1);
                while(pos2  !=  string::npos){
                        temp.erase(pos2,  temp1.length());
                        pos2  =  temp.find(temp1);
                }
                cout  <<  endl;
                temp1  =  Find_Repeat_String(temp);
        }
        int  minlen  =  9999;
        for(int  i  =  0  ;  i  <  vs.size()  ;  i++){
                string  s  =  vs[i];
                if(s.length()  <  minlen){
                        minlen  =  s.length();
                }
        }
        //found  the  gcd  of  key  length  is  3
        cout  <<  "GCD:  "  <<  minlen  <<endl;
        cout  <<  "..................................................
........"  <<  endl;
```

第二步：密钥长度为 3，找到了密钥长度后，对加密字符串进行分组，分为 3 组，比如：abcdefg，第一组为 adg，第二组为 be，第三组为 cf 等，然后再在每个组找出哪个字符出现的频率较高，对它们进行频率分析。




```
//get the frequency
        //minlen is 3 then get the char that separate with 3 char
        //ex : abcde... , then we will get ad,be,c...
        for(int i = 0 ; i < minlen ; i++){
                string s = "";
                for(int j = i ;j<str.length();j = j + minlen){
                        s = s + str[j];
                }
                cout << "第" << i+1 << "组 : " << s << endl;
                int count[26] = {};
                for(int j = 0 ; j < s.length() ; j++){
                        int index = s[j] - 'a';
                        count[index]++;
                }
                int maxcount = 0;
                for(int j = 0 ; j < 26 ; j++){
                        if(count[j] > count[maxcount])
                                maxcount=j;//get the maximum count index
                        if(count[j] != 0)
                                cout << char(j + 'a') << " : " << count[j
] << endl;
                }
                cout << "Maximum occurrence of a character : " << char(ma
xcount + 'a') << endl;
                cout << "Number of Occurrence : " << count[maxcount] << "
times" << endl;
                cout << "..........................................................
......" << endl;
        }
```

第三步：我们可从图中分析出有哪些密文是有可能与明文出现重叠的字符，这可表示其中一个密钥有可能会是'a'这个字符。由图可知，比如 p, e, s 这三个字符是有可能的（看图可猜测），然后在 16 种组合中的每个字符都分别减去 p 或 e 或 s 这三个字符再加上'a'来得到有可能形成密钥的组合。之后，把原始字符串取出来，把 48 种有可能是密钥的字符串与原始字符串进行操作，把每个原始字符与密钥的相对应字符进行相减再加上“a”，比如密钥是“con”，原始字符“ofsgrw”，“o”-“c”得出字符…，“f”-“o”得字符…，“s”-“n”得出…，“g”-“c”得…，以此类推。进行完这一系列操作后，再分析与识别是哪一个密钥解出的字符串是属于明文。如果有一组的猜测密钥可以显示出明文的话，那就代表找到密钥了。

```
//find the possible key to decrypt vignere cipher
        string combination[16] = {"cex","cox","csx","ctx","gex","gox","gsx","gtx
",
                                                                "pex","pox","psx","
ptx","vex","vox","vsx","vtx"};
        string possible_key[48] = "";
        int k = 0, countkey = 0;
        char c[5] = "pes";
        while(k != 3){
                string temp[16] = combination;
                for(int i = 0 ; i < 16 ; i++){
                        for(int j = 0 ; j < 3 ; j++){
                                temp[i][j] = (temp[i][j] - c[k] + 26 ) % 2
6 + 'a';
                        }
                        possible_key[countkey] = temp[i];
                        countkey++;
                }
                k++;
        }
        for(int i = 0 ; i < 48 ; i++){
                string s = str;
                for(int j = 0 ; j < s.length() ; j++){
                        s[j] = ( (s[j] - 'a') - (possible_key[i][j % 3] -
 'a') + 26 ) % 26 + 'a';
                }
                cout << i+1 << "." << "可能是密文得密钥:
" << possible_key[i] << endl;
                cout << s << endl ;
        }
```

```
21.可能是密文的密钥: cat
itisessentialtoseekoutenemyagentswhohavecometoconductespionageagainstyouandtobribethemtoserveyougivetheminstructionsandc
areforthemthusdoubledagentsarerecruitedandusedsuntzutheartofwar
```

密钥：cat

明文：it is essential to seek out enemy agents who have come to conduct espionage against you and to bribe them to server you give them

instructions and care for them thus doubled agents are recruited and used sun tzu the art of war

密文 3：未知密码

Unknown Ciphertext : MAL TIRRUEZF CR MAL RKZYIOL EX MAL OIY UAE RICF “MAL ACWALRM DYEUPLFWL CR ME DYEU MAIM UL IZL RKZZEKYFLF GH OHRMLZH”

解密方式：MAL 字符串频繁出现在句子中，可判断为句子中常出现的语句助词之类的。我起初假设这个单词是“the”，然后再去列表来判断。我发现在句子中没有以下几个字符 BJNQSV。

MAIM:THAT , I=A

IZL:ARE , Z=R

ME:TO , E=O

UAE:WHO , U=W , 所以 UL=WE , IZL=ARE

EX:OF , X=F

CR=IS , C=I , R=S

TIRRUEZF=PASSWORD , T=P , F=D , 所以 RICF=SAID

ACWALRM=HIGHEST , W=G

DYEU=__OW , 经过我的猜测应该是 IS TO KNOW THAT 的 KNOW , DYEU=KNOW , D=K Y=N

DYEUPLFWL=KNOWLEDGE , P=L

OIY=MAN , O=M

RKZYIOL=SURNAME , K=U , 所以 RKZZEKYFLF=SURROUNDED

GH=BY , G=B , H=Y , 所以 OHRMLZH=MYSTERY

下表是我分析出来的明文与密文的关系：

密钥：

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	空	I	K	O	D	B	Y	A	空	U	E	T	空	M	L	空	S	空	P	W	空	G	F	N	R

明文：THE PASSWORD IS THE SURNAME OF THE MAN WHO SAID THE HIGHEST KNOWLEDGE IS TO KNOW THAT WE ARE SURROUNDED BY MYSTERY.

所以那位说过这句话的人是 Albert Schweitzer。

3190300677 姚熙源