《实验一 加解密算法的实现》实验报告

姓名	李宽宇	年级	21 级
学号	20215279	专业、班级	21 计卓 1 班

实验名称	实验一 加解密算法的实现		
实验时间	2024.3.30	实验地点	DS3305
实验成绩		实验性质	□验证性 □设计性 □综合性

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

掌握频度分析法原理和 Feistel 加解密原理

使用频度分析法解密文本，并写出替换表

编程实现 Feistel 加密/解密

二、实验项目内容

1.使用频度分析法解密以下文本，并给出替换表：

UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX AIZ VUEPHZ HMDZSHZO WSFP APPD TSVP QUZW YMXUZUHSX EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD TMOHMQ

2.编程实现 Feistel 加密解密以下文本：

CQUINFORMATIONSECURITYEXP

三、实验设计

（一）实验原理、原理图

1.频度分析原理

对于任何一种书面语言而言，不同的字母或字母组合出现的频率各不相同。如果以这种语言书写足够长的文本，都呈现出大致相同的特征字母分布规律。

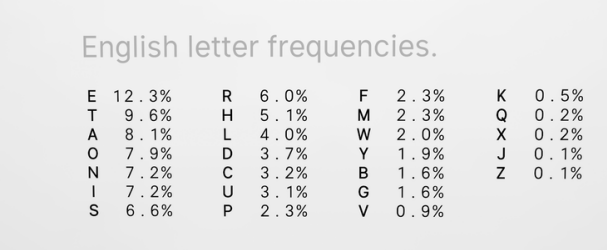


图 1：英文单字母统计特性

在上表中，不少字母出现的概率近乎相等，但也有极少数字母出现的概率有较大差异。为了分析方便密文信息，常将英文字母表按字母出现的概率大小分类，分类情况如下：

极高频	E
次高频	T A O I N S H R
中等频	D L
低频	C U M W F G Y P B
甚低频	V K J X Q Z

表 1 单字母统计特性频率表

语言的**单字母统计特性**没有反映出英文**双字母和多字母的特征**，在双字母中统计出概率最大的 30 对字母按概率大小排列为：

th he in er an re ed on es st en at to nt ha nd ou ea ng as or ti is et it ar te se hi of

类似的，我们还可以考察英文课文中**三字母出现的频率**。按 Beker 在 1982 年统计的结果(样本总数 100 360)得到概率最大的 20 组三字母按概率大小排列为：

the ing and her ere ent tha nth was eth for dth hat she ionhis sth ers ver

特别地，the 出现的频率几乎为 ing 的 3 倍。

2. Feistel 加密/解密原理

参考

[密码学系列之:feistel cipher - 知乎 (zhihu.com)]
[https://blog.csdn.net/Drifter_Galaxy/article/details/107702084]

Feistel 加密/解密原理如下图所示，

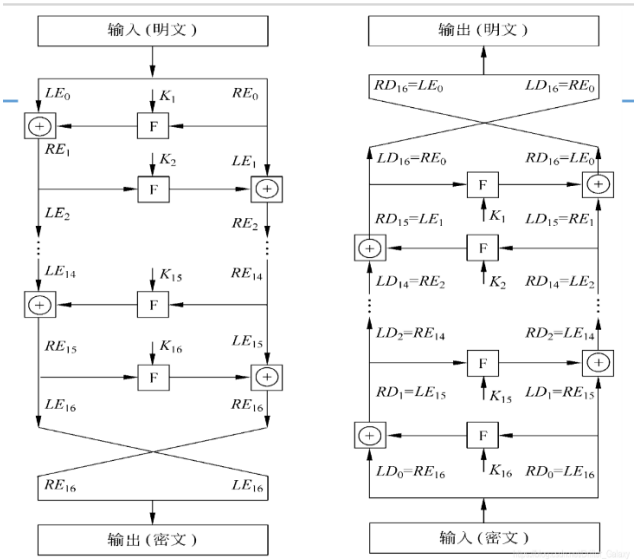


图 2 Feistel 加密/解密原理图

Feistel 网络中会用到一个 round function（F），这个函数接收两个输入参数，分别是分组数据和 key，然后生成和分组数据同样长度的数据。

首先使用上一轮生成的数据和原始数据的另一半进行 XOR 异或操作，作为下一轮轮函数的输入。

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F\left(R_{i-1},\ K_i\right)$$

就这样一轮一轮进行下去最后生成加密过后的数据。

解密的流程和加密的流程是类似的，只不过把加密的操作反过来。Feistel 网络的轮数可以任意增加。不论多少轮都可以正常解密。解密与轮函数 f 无关，轮函数 f 也不需要有逆函数。轮函数可以设计得足够复制。

四、实验过程或算法(关键步骤、核心代码注解等)

（一）使用频度分析法解密以下文本，并给出替换表：

UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX AIZ VUEPHZ HMDZSHZO WSFP APPD TSVP QUZW YMXUZUHSX EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD TMOHMQ

1.假定给定文本采用了简单的代换加密。

2.统计单字符频度：对加密文本中的字符进行频度分析。计算每个字符出现的次数，并记录下来。

明文	频率
p	0.1333
z	0.1167
s	0.0833
u	0.0833
o	0.075
m	0.0667
h	0.0583
e	0.05
d	0.05
x	0.0417
v	0.0417
w	0.0333
f	0.0333
q	0.025
t	0.025
b	0.0167
a	0.0167
g	0.0167
y	0.0167
i	0.0083
j	0.0083

表 2：单字符频度

3.统计双字符字符频度

出现次数	双字符
1	mq ym zp mb uh mx vp ap vs sf me et zo pz ai oz ye pp xm hx tm dj up fu uf bm xu db ph es qu pf sh ev uo iz sg ue so sv hs pv gp qs py wp
2	vu md dz zu oh ts hz mo ws sz zs pd ud pe
3	op zw sx po ep uz hm fp

表 3：双字符频度

出现次数	三字符
1	mdz gpo phz eph zuh upz mxu poz qso hzo mop pev ets shz wsf pop fup pye epo ppd yep app uep pdz uhs szu xmo xuz ymx uzu uzw quz hsx svp sfp tsx zws zpe aiz mdj fpe pfp esx vue bme vsg uoh tsv opd zsz zuf tmo moh ohm opf ozp pes udb met dbm hxm fpo epy hmq wsz ufp zwp vuo evs opv ohx zsh
2	dzs hmd

表 4：三字符频度

统计字符的源代码，先转大小写，统计频数，排序，结果写到.csv

```cpp
bool compareByFrequency(const pair<char, int>& a, const pair<char, int>& b) {
    return a.second > b.second;  // 按照出现次数降序排序
}

int main() {
    string text="UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX AIZ VUEPHZ HMDZSHZO WSFP APPD TSVP QUZW YMXUZUHSX EPYEPOPDZSZUFPO MB ZWP F
    unordered_map<char, int> charFrequency;

    for (char c : text) {
        if (isalpha(c)) {  // 只考虑字母字符
            charFrequency[tolower(c)]++;  // 不区分大小写，将字母转换为小写
        }
    }
    vector<pair<char, int>> sortedChars(charFrequency.begin(), charFrequency.end());
    sort(sortedChars.begin(), sortedChars.end(), compareByFrequency);
    ofstream outputFile("output.csv");
    if (outputFile.is_open()) {
        outputFile << "字符,出现频数,出现频率" << endl;
        double sum = 0;
        for (const auto& pair : sortedChars) {
            sum += pair.second;
        }
        for (const auto& pair : sortedChars) {
            outputFile << pair.first << "," << pair.second << "," << fixed << setprecision(4) << pair.second / sum << endl;
        }
        outputFile.close();
        cout << "数据已成功写入 output.csv 文件中。" << endl;
    } else {
        cout << "无法打开文件 output.csv" << endl;
    }

    return 0;
}
```

图 3：单字符统计

统计双字符的源代码，把 char 换成 string 即可

图 4：双字符统计

4. 比较频度：将字符频度与英文语言中常见字符的频度进行比较。

	明文	密文
极高频	p	E
次高频	z, s, u, o, m, h, e, d	T A O I N S H R
中等频	x, v,	D L
低频	w, f, q, t, b, a, g, y, i	C U M W F G Y P B
甚低频	j	V K J X Q Z

表 5：单字符频度比较

5. 尝试解密：根据频度分析的结果，尝试使用最可能的映射关系对加密文本进行解密。

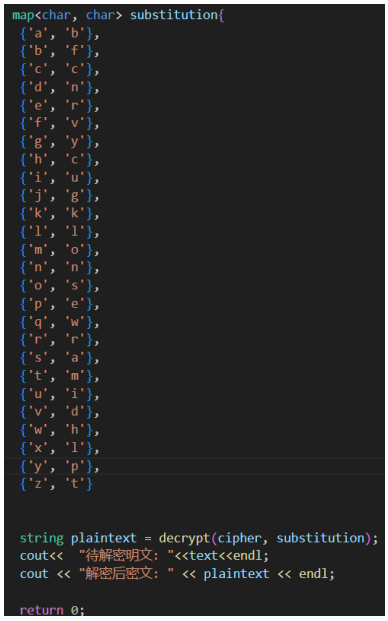
首先，根据 3 字符频度分析， 先假设 p->e

解密映射源代码：

```
// 解密函数
string decrypt(const string &cipher, const map<char, char> &substitution)
    string plaintext;
    for (char c : cipher) {
        if (substitution.count(c) > 0) {
            plaintext += substitution.at(c);
        } else {
            plaintext += c; // 如果字符不在代换规则中，则保留原字符
        }
    }
    return plaintext;
}
```

图 5 解密代换函数

解密映射源的 map，并输出结果：



```
map<char, char> substitution{
 {'a', 'b'},
 {'b', 'f'},
 {'c', 'c'},
 {'d', 'n'},
 {'e', 'r'},
 {'f', 'v'},
 {'g', 'y'},
 {'h', 'c'},
 {'i', 'u'},
 {'j', 'g'},
 {'k', 'k'},
 {'l', 'l'},
 {'m', 'o'},
 {'n', 'n'},
 {'o', 's'},
 {'p', 'e'},
 {'q', 'w'},
 {'r', 'r'},
 {'s', 'a'},
 {'t', 'm'},
 {'u', 'i'},
 {'v', 'd'},
 {'w', 'h'},
 {'x', 'l'},
 {'y', 'p'},
 {'z', 't'}


string plaintext = decrypt(cipher, substitution);
cout<<  "待解密明文: "<<text<<endl;
cout << "解密后密文: " << plaintext << endl;

return 0;
```

图 6 代换 map 与输出

6. 逐步调整：逐步调整映射关系，不断尝试解密文本，直到整个文本都被解密为止。

注意到 UZ,MB，UD 是两个字符的单词，结合已知常见单词可能情况如下：

UZ -> it an he

MB/UD->in on as at or of it is he an

U !-> h

U->i/o

推测出 u->i ,z->t,d->n

再根据三到四字符单词 ZWP、ZWSZ、QUZW 推测出如下结论：

ZWP -> the

ZWSZ ->that

QUZW->with

此时，

待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it wao viohxmoev geoteevag that oefeeax inbmetax ait vieeht hmntahto hafe aeen tave with ymxitihax

eeyeeoentatifeo mb the fiet hmnj in tmohmw

观察单词 Qso、MB：

其中 qso-> wao ，结合常出现三字符单词（如下列出）,推测为 was

the ing and her ere ent tha nth was eth for
dth hat she ionhis sth ers ver

MB - > as at or of is

其中 wsfp appd - > hafe aeen 推测为 have been

推测出 f->v, a-b

此时，

待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx
aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx
epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was vishxmsev gesteevag that seveeax inbmetax
bit vieeht hmntahts have been tave with ymxitihax
eeyeesentatives mb the viet hmnj in tmshmw

根据 Opfpesx->seveeax 推测 several

推测出 e->r,x->l

此时，

待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx
aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx
epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was vishlmsev gestervag that several inbmrtal
bit vireht hmntahts have been tave with ymlitihal
reyresentatives mb the viet hmnj in tmshmw

根据 vuephz -> vireht 推测 direct

推测出 v->d,h->c

此时，

待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx
aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx
epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was disclmsed gesterdag that several inbmrtal
bit direct cmntacts have been tade with ymlitical

reyresentatives mb the viet cmnj in tmscmw

vuohxmopv -> disclmsed 推测 disclosed

m->o

appd tsvp quzw - > been tade with,’ tade’ 应该是过去分词，结合频率表推测

t->m

此时，

待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was disclosed gesterdag that several inbormal bit direct contacts have been made with yolitical reyresentatives ob the viet conj in moscow

根据 udbmetsx - > inbormal 推测 informal，

且 mb->ob ,mb 范围在 in on as at or of it is he an

低频	w,f,q,t,b,a,g,y,i	C U M W F G Y P B

所以 b->f

此时，

带解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx epyepopdzszufpo mb zwp fupz hmdj ud tmohmq
解密后密文: it was disclosed gesterdag that several informal bit direct contacts have been made with yolitical reyresentatives of the viet conj in moscow

仍有 5 个单词没有猜出来

根据 epyepopdzszufpo-> reyresentatives 推测是 representatives

推测出 y->p

根据 viet conj -> viet cong

推测出 j->g

此时，

待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was disclosed gesterdag that several informal bit direct contacts have been made with political

representatives of the viet cong in moscow

根据 informal bit direct 两个形容词只能用 but 连接

推测出 i->u

根据 gesterdag 推测 yesterday

推测出 g->y

此时，待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow

7.验证解密结果：解密完成后，验证解密结果是否合理和通顺。

翻译成中文：昨天有消息透露，与在莫斯科的越共政治代表进行了几次非正式但直接的接触

得到替换表

明文	a	b	c	d	e	f	g	h
密文	b	f		n		v	y	c
明文	i	j	k	l	m	n	o	p
密文	u	g			o		s	e
明文	q	r	s	t	u	v	w	x
密文	w		a	m	i	d	h	
明文	y	z						
密文	p	t						

表 6：替换表

(二)编程实现 Feistel 加密解密以下文本：

CQUINFORMATIONSECURITYEXP

1.长度不是偶数，末尾补上\032(空格)，用 vector<char>存储，string 会出现许多不适合打印或无法存储的符号，如\n,\0(null)

2.确定轮次，并随机生成 32 位密钥

```cpp
const int rounds = 16;            // 轮数
// 随机生成密钥
vector<unsigned int> generateRandomKeys() {
    vector<unsigned int> keys;
    srand(time(0));
    for (int i = 0; i < rounds; ++i) {
        unsigned int key = rand();
        keys.push_back(key);
    }
    return keys;
}
// 固定密钥
// vector<unsigned int> generateRandomKeys() {
//     vector<unsigned int> keys;
//     srand(time(0));
//     for (int i = 0; i < rounds; ++i) {
//         keys.push_back(12345678);
//     }
//     return keys;
// }
vector<unsigned int> keys = generateRandomKeys();
```

图 7：固定密钥和随机密钥生成的代码

3. 轮转函数结果完成与 L 的异或操作

```cpp
vector<char> FxorStrings(vector<char> R, vector<char> L,size_t round)
{
    vector<char> result;
    for (size_t i = 0; i < L.size(); ++i)
    {
        // unsigned int 32 位每次取8位
        // Example Feistel round operation: XOR with key
        char temp = R[i] ^ (keys[round] >> (i % 4 * 8));
        result.push_back(L[i] ^ temp);
    }
    return result;
}
```

图 8：轮转函数结果完成与 L 的异或操作代码

4. 单个 round

```cpp
pair<vector<char>, vector<char>> feistelOneRound(vector<char> L, vector<char> R,size_t round)
{
    pair<vector<char>, vector<char>> result;
    result.second = R;
    result.first = FxorStrings(R, L,round);
    return result;
}
```

图 9：单个 round 代码

5. 多个 round, 当解码时，key 要倒序

```cpp
pair<vector<char>, vector<char>> feistel(vector<char> L, vector<char> R,bool decode=false)
{
    if(decode){
        reverse(keys.begin(), keys.end());
    }
    pair<vector<char>, vector<char>> result;
    for (size_t i = 0; i < rounds; ++i)
    {
        result = feistelOneRound(L, R,i);
        L = result.second;
        R = result.first;
    }
    return result;
}
```

图 10：多个 round 代码

7. 输出结果如下

```
加密前
67 81 85 73 78 70 79 82 77 65 84 73 79 78 83 69 67 85 82 73 84 89 69 88 80 32
加密
-43 115 16 10 -61 101 6 6 -52 117 12 25 -73 95 63 69 67 68 62 73 84 72 41 88 80 49
解密:
67 81 85 73 78 70 79 82 77 65 84 73 79 78 83 69 67 85 82 73 84 89 69 88 80 32
此时的keys
13894 18559 29101 10072 12671 2068 13116 31599 16501 2577 15388 27802 20736 8019 6247 10699
```

图 11：控制台结果截屏

五、实验过程中遇到的问题及解决情况(主要问题及解决情况)

1. 主要问题：统计了单字符、双字符、三字符后，仍然难以对应代换

解决情况：转变思路，从二字符、三字符单词出发，根据常见单词进行推测。

2. 主要问题：实验 2 中，Feistel 加密和解密的结构是一样的，而忽视了 key 的顺序是颠倒的

解决情况：

```
if(decode){
    reverse(keys.begin(), keys.end());
}
```

图 12：解码颠倒 keys 的顺序

3. 主要问题：实验 2 中,char 是 8 位的，但是密钥是 32 位的 int

解决情况：轮流 int,每次移 8 位

4. 主要问题：实验 2 中,固定的密钥会使部分密文出现在明文中

```
加密前
67 81 85 73 78 70 79 82 77 65 84 73 79 78 83 69 67 85 82 73 84 89 69 88 80 32
加密
67 99 -84 10 85 117 -70 6 90 101 -80 25 33 78 83 69 67 85 82 73 84 89 69 88 80 32
解密:
67 81 85 73 78 70 79 82 77 65 84 73 79 78 83 69 67 85 82 73 84 89 69 88 80 32
```

图 13：固定的密钥会使部分密文出现在明文中控制台结果截屏

解决情况：替换成随机生成密钥

六、实验结果及分析和（或）源程序调试过程

结果分析

（一）待解密明文: uz qso vuohxmopv gpozpevsg zwsz opfpesx udbmetsx aiz vuephz hmdzshzo wsfp appd tsvp quzw ymxuzuhsx epyepopdzszufpo mb zwp fupz hmdj ud tmohmq

解密后密文: it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow

验证解密结果：解密完成后，验证解密结果是否合理和通顺。

翻译成中文：昨天有消息透露，与在莫斯科的越共政治代表进行了几次非正式但直接的接触

得到替换表

明文	a	b	c	d	e	f	g	h
密文	b	f		n		v	y	c
明文	i	j	k	l	m	n	o	p
密文	u	g			o		s	e
明文	q	r	s	t	u	v	w	x
密文	w		a	m	i	d	h	
明文	y	z						
密文	p	t						

表 6：替换表

（二）编程实现 Feistel 加密解密以下文本：

CQUINFORMATIONSECURITYEXP

```
加密前
67 81 85 73 78 70 79 82 77 65 84 73 79 78 83 69 67 85 82 73 84 89 69 88 80 32
加密
-43 115 16 10 -61 101 6 6 -52 117 12 25 -73 95 63 69 67 68 62 73 84 72 41 88 80 49
解密:
67 81 85 73 78 70 79 82 77 65 84 73 79 78 83 69 67 85 82 73 84 89 69 88 80 32
此时的keys
13894 18559 29101 10072 12671 2068 13116 31599 16501 2577 15388 27802 20736 8019 6247 10699
```

图 14：控制台结果截屏

密文：-43 115 16 10 -61 101 6 6 -52 117 12 25 -73 95 63 69 67 68 62 73 84 72 41 88 80 49

Keys: 13894 18559 29101 10072 12671 2068 13116 31599 16501 2577 15388 27802 20736 8019 6247 10699