实验一报告

一 移位密码

移位密码：将英文字母向前或向后移动一个固定位置。例如向后移动3个位置，即对字母表作置换（不分大小写）。

```
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

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

设明文为：`public keys`,则经过以上置换就变成了：`sxeolf nhbv`。

如果将26个英文字母进行编码：$A \to 0, B \to 1, \cdots, Z \to 25$，则以上加密过程可简单地写成：

- 明文：$m = m_1 m_2 \cdots m_n$,
- 密文：$c = c_1 c_2, \cdots c_n$，其中$c_i \equiv (m_i + key) \pmod{26}, i = 1, 2, \cdots, n.$

C++实现原理

```cpp
#include <algorithm>
#include <cstdint>
#include <iostream>
#include <string>

#include "utils.hh"
int main(int argc, const char** argv)
{
    std::cout << "Enter string: " << std::endl;
    std::string input;
    getline(std::cin, input);

    const uint32_t distance = random_distance();

    std::transform(input.begin(), input.end(), input.begin(), [distance](char c) -> char {
        if (c != ' ') {
            return (char)(((std::tolower(c) - 'a' + distance) % 25 + 'a'));
        } else {
            return c;
```

```
        }
    });

    std::cout << "Result: " << std::endl;
    std::cout << input << std::endl;

    return 0;
}
```

其工作原理可以概括如下：

1. 首先根据`Mersenne Twister`随机数种子均匀地生成一个随机密钥 $k \leftarrow \text{UniformRandom}(0, 25)$：

```
uint32_t random_distance(void)
{
    std::random_device rd;
    std::mt19937 engine { rd() };
    std::uniform_int_distribution<uint32_t> dist(0, 25);
    return dist(engine);
}
```

2. 根据输入的字符串，先转换成小写字母，然后再根据公式进行代换：

$$c_i \equiv m_i + k \pmod{26}.$$

移位密码的攻击手段

很明显，这个密钥空间只有26，所以我们直接穷举即可，输出26个可能的的字符串即可。这个过程和上述加密过程是对称的：

```
    for (uint32_t i = 0; i < 26; i ++) {
        std::string guess;
        std::transform(input.begin(), input.end(), std::back_inserter(guess),
[i](const char& c) -> char {
            if (c == ' ') {
                return c;
            } else {
                return (char)(((std::tolower(c) - 'a' + i) % 25 + 'a'));
            }
        });

        std::cout << "Guess " << i << ": " << guess << std::endl;
    }
```

二 单表置换密码

单表置换密码就是根据字母表的置换对明文进行变换的方法，例如，给定置换

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

那么给定明文`public keys`，则有密文：`mckebw qxuo`。单表置换实现的一个关键问题是关于置换表的构造。置换表的构造可以有各种不同的途径，主要考虑的是记忆的方便。如使用一个短语或句子，删去其中的重复部分，作为置换表的前面的部分，然后把没有用到的字母按字母表的顺序依次放入置换表中。

我们可以定义一个带密钥的伪随机置换函数PRP来做到这个置换密码，$\text{arr}' \leftarrow \text{PRP}_k(\text{arr})$，其中$k \leftarrow \text{KeyGen}(1^\lambda)$（$\lambda$为安全参数）。

C++实现

```
#include <iostream>
#include <string>
#include <random>
#include <vector>

static std::string substitution_table = "abcdefghijklmnopqrstuvwxyz";

int main(int argc, const char** argv)
{
```

```
    std::cout << "Enter string: " << std::endl;
    std::string str;
    getline(std::cin, str);

    // Do the permutation.
    std::random_device rd;
    std::mt19937 engine { rd() };
    std::shuffle(substitution_table.begin(), substitution_table.end(),
engine);

    std::transform(str.begin(), str.end(), str.begin(), [](char c) -> char {
        if (c != ' ') {
            return substitution_table[std::tolower(c) - 'a'];
        } else {
            return c;
        }
    });

    std::cout << "Result: " << std::endl << str << std::endl;
    return 0;
}
```

样例输出

```
Enter string:
jsdiajdisjaijdsaiisdaiasd

Guess 0: jsdiajdisjaijdsaiisdaiasd
Guess 1: ktejbkejtkbjketbjjtebjbte
Guess 2: lufkclfkulcklfuckkufckcuf
Guess 3: mvgldmglvmdlmgvdllvgdldvg
Guess 4: nwhmenhmwnemnhwemmwhemewh
Guess 5: oxinfoinxofnoixfnnxifnfxi
Guess 6: pyjogpjoypgopjygooyjgogyj
Guess 7: qakphqkpaqhpqkahppakhphak
Guess 8: rblqirlqbriqrlbiqqbliqibl
Guess 9: scmrjsmrcsjrsmcjrrcmjrjcm
Guess 10: tdnsktnsdtkstndkssdnkskdn
Guess 11: ueotluoteultuoeltteoltleo
Guess 12: vfpumvpufvmuvpfmuufpmumfp
Guess 13: wgqvnwqvgwnvwqgnvvgqnvngq
Guess 14: xhrwoxrwhxowxrhowwhrowohr
```

```
Guess 15: yisxpysxiypxysipxxispxpis
Guess 16: ajtyqatyjaqyatjqyyjtqyqjt
Guess 17: bkuarbuakbrabukraakurarku
Guess 18: clvbscvblcsbcvlsbblvsbslv
Guess 19: dmwctdwcmdtcdwmtccmwtctmw
Guess 20: enxduexdneudexnuddnxudunx
Guess 21: foyevfyeofvefyoveeoyvevoy
Guess 22: gpafwgafpgwfgapwffpawfwpa
Guess 23: hqbgxhbgqhxghbqxggqbxgxqb
Guess 24: irchyichriyhicryhhrcyhyrc
Guess 25: jsdiajdisjaijdsaiisdaiasd
```

三 频率分析来破译单表置换密码

在单表置换密码中，由于置换表字母组合方式有$26!$种，约为4.03×10^{26}种组合，所以采用穷举密钥的方法不是一种最有效的方法。对单表置换密码最有效的攻击方法是利用自然语言的使用频率：单字母、双字母组/三字母组、短语、词头/词尾等。我们考虑英文。

短单词(small words):在英文中只有很少几个非常短的单词。因此，如果在一个加密的文本中可以确定单词的范围，那么就能得出明显的结果。一个字母的单词只有a和I。如果不计单词的缩写，在从电子邮件中选取500k字节的样本中，只有两个字母的单词仅出现35次，而两个字母的所有组合为26×26＝676种。而且，还是在那个样本中，只有三个字母的单词出现196次，而三个字母的所有组合为26×26×26＝17576种。

常用单词(common words):再次分析500k字节的样本，总共有5000多个不同的单词出现。在这里，9个最常用的单词出现的总次数占总单词数的21%，20个最常用的单词出现的总次数占总单词数的30%，104个最常用的单词占50%，247个最常用的单词占60%。样本中最常用的9个单词占总词数的百分比为：

```
   the 4.65   to  3.02  of  2.61  I  2.2  a  1.95

   and 1.82   is  1.68  that 1.62  in  1.57
```

字母频率(character frequency):在1M字节旧的电子文本中，对字母"A"到"Z"（忽略大小写）分别进行统计。发现近似频率（以百分比表示）：

```
e 11.67  t 9.53  o 8.22  i 7.81  a 7.73  n 6.71 s 6.55

r 5.97  h 4.52  l 4.3  d 3.24  u 3.21  c 3.06 m 2.8

p 2.34  y 2.22  f 2.14  g 2.00  w 1.69  b 1.58 v 1.03

k 0.79  x 0.30  j 0.23  q 0.12  z 0.09
```

从该表中可以看出，最常用的单字母英文是e和t，其他字母使用频率相对来说就小得多。这样，攻击一个单表置换密码，首先统计密文中最常出现的字母，并据此猜出两个最常用的字母，并根据英文统计的其他特征（如字母组合等）进行试译。

破译一段密文

```
SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJHAY JBRCGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QCRRNEC GNB MBAY HC PCGMTCPCD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJXNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QCRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRICR SM ENJB ZBNZSIMPJOCD GMBSPMA MF SIC QCRRNEC
```

1. 先编写一段朴素的C++程序来统计这个语言中每个字母的出现频率：

```cpp
// Count occurrence.
uint32_t total = 0;
std::for_each(cipher.begin(), cipher.end(), [&freq, &total](const char& c) {
    if (std::isalpha(c)) {
        freq[c - 'a']++;
        total++;
    }
});
// Compute frequency.
std::vector<std::pair<char, double>> freq_cipher; // Frequency of the cipher.
char c = 'a';
std::transform(freq.begin(), freq.end(), std::back_inserter(freq_cipher), [total, &c](const uint32_t& count) {
    std::cout << c << ", " << count * 1.0 / total << std::endl;
    return std::make_pair(c++, count * 1.0 / total);
});
```

```
// Sort the frequency table.
std::sort(freq_cipher.begin(), freq_cipher.end(),
    [](const std::pair<char, double>& lhs, const std::pair<char, double>& rhs) -> bool {
        return lhs.second > rhs.second;
    });
```

经过上述的统计可以初步获取每个字母的出现频次排序为：

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

初步拟合的结果为：

```
the leatsou dsimuep na lsydticsodhy nr thot if tsoarpnttnac nafispotnia fsip o dinat o ti o dinat m my peoar if o dirrnmuy narelgse lhoaaeu na rglh o boy thot the isncnaou perroce loa iauy me selivesew my the snchtfgu selndneatr the dostnlndoatr na the tsoaroltnia ose ounle the isncnaotis if the perroce mim the selenves oaw irlos o dirrnmue iddiaeat bhi bnrher ti cona gaogthisnkew liatsiu if the perroce
```

已经有一些词汇被正确识别了，接下来分析还没有分析正确的词汇。

2. 注意单个字母的单词只有a和I，因此单个字母的单词只能根据频率匹配这两个单词，故o匹配a，所以再次精细化得到结果：

```
the leatsau dsimuep na lsydticsadhy nr that if tsaarpnttnac nafispatnia fsip a dinat a ti a dinat m my peaar if a dirrnmuy narelgse lhaaaeu na rglh a bay that the isncnaau perrace laa iauy me selivesew my the snchtfgu selndneatr the dastnlndaatr na the tsaaraltnia ase aunle the isncnaatis if the perrace mim the selenves aaw irlas a dirrnmue iddiaeat bhi bnrher ti cana gaagthisnkew liatsiu if the perrace
```

3. 接下来分析两个字母的单词：

```
if:3 na:3 nr:1 ti:2 my:2 me:1
```

两个字母的单词也比较少，所以根据频率和逻辑可以推测

```
na -> in
if -> of
ti -> to
nr -> is
my -> by
```

可以再次精细化结果得到

```
the lentsau dsobuep in lsydticsadhy is that if tsanspittinc infospation fsop a doint a to a doint b by peans of a dossibuy inselgse lhanneu in sglh a bay that the osicinau pessace lan onuy be selovesew by the richtfgu selndients the dastilidants in the tsaasaltion ase auile the osiciaatos of the pessace bob the seleives aaw oslas a dossimue oddonent bhi bishes to cain gaagthosikew liatsou of the pessace
```

4. 此时很多文字已经可以辨认了，例

```
bishes -> wishes b -> w
infospation -> information s -> r, p ->m, c->g
```

继续恢复：

```
the lentrau drobuep in lrydtogradhy is that if transmitting information from a doint a to a doint b by means of a dossibuy inselgre lhanneu in sglh a way that the originau message lan onuy be reloverew by the rightfgu relidients the dartilidants in the transaction are auile the originator of the message bob the releiver arw oslar a dossnmue oddonent who wishes to gain gnagthirikew lontrou of the message
```

5. 接下来可以继续认清单词：

```
onuy -> only   u -> l
rightfgu -> rightful  g -> u
dossibuy -> possibly  d -> p
```

继续恢复：

```
the lentral problep in lryptography is that if transmitting information from a point a to a point b by means of a possibly inselgre channel in sglh a way that the original message lan only be reloverew by the rightful relipients the partilipants in the transaction are alile the originator of the message bob the releiver arw oslar a possimle opponent who wishes to gain gnagthorikew lontrol of the message
```

6. 基本上就已经清楚了：

```
lcryptography -> cryptography   l -> c
problp -> problem   p -> m
relocerew ->recovered  w -> d
```

```
the central problem in cryptography is that if transmitting information from a point a to a point b by means of a possibly insecgre channel in sgch a way that the original message can only be recovered by the rightful recipients the participants in the transaction are alice the originator of the message bob the receiver ard oscar a possimle opponent who wishes to gain gnagthoriked control of the message
```

7. 最后有

```
insecgre -> insecure g -> u
possimle -> possible m -> b
unauthoriked -> unauthorized k -> z
```

```
the central problem in cryptography is that if transmitting information from a point a to a point b by means of a possibly insecure channel in such a way that the original message can only be recovered by the rightful recipients the participants in the transaction are alice the originator of the message bob the receiver ard oscar a possimle opponent who wishes to gain unauthorized control of the message.
```