Caesar密码

1 加密原理

凯撒密码是一种替换加密技术，明文中的所有字母都在字母表上向后（或向前）按照一个固定数目进行偏移后被替换成密文。

2 破解过程

由于字符串的偏移量相同，可以通过枚举25种偏移方式来暴力破解。

```
for (int i = 1; i < 26; i++) {
    for (int j = 0; input[j] != 0; j++) {
      input[j] = (input[j] + 1 - 'A') % 26 + 'A';
    }
}
```

输出25种可能的结果后，注意到`pleaseencryptyournamewiththesamekeyanduploadtolearninginzju`有实际含义。因此，加密时偏移量为`-10`。

Vigenere密码

1 加密原理

维吉尼亚密码是由一些偏移量不同的恺撒密码组成，通过密钥来计算对应明文的位移量。

2 破解过程

2.1 重合指数计算

在一段字符串中，重合指数的值为：

$$\kappa = \frac{\sum_{i=1}^{c} n_i(n_i - 1)}{N(N-1)}$$

其中，c是指字母表的长度（英文为26），N指文本的长度，n_1到n_c是指密文的字母频率，为整数。

通过对密钥长度的枚举，获得不同长度的字串，当重合指数的平均值处于[0.06, 0.07]之间时，可以认为是正确的密钥长度。

经过计算，注意到当密钥长度`l=3`时，重合指数符合上述区间。

2.2 拟重合指数计算

将密文拆分为3个子串，分别枚举它们的偏移量（密钥），计算拟重合指数，25个偏移量中拟重合指数最大者为正确密钥。

拟重合指数的值为：

$$\chi = \sum_{i=1}^{n} p_i q_i$$

其中p_i为字母出现的概率，q_i为字母在字符串中出现的频率。

比较之后最终得出，密钥为`CAT`，明文为

```
it is essential to seek out enemy agents who have come to conduct espionage
against you and to bribe them to serve you give them instruction sand care for
them thus doubled agents are recruited and used suntzu the art of war
```

Unknown

1 破解过程

首先假设密文字母与明文字母一一对应。

其次对字符串各字母的出现频率进行统计，注意到密文字母`L`，`M`和`R`出现频率最高，同时发现MAL反复出现，对照英文字母出现概率猜测`M`和`L`分别对应`T`和`E`。推出`A`对应`H`。

通过词义和字母频率的多次对照后，可以最终得出答案`the password is the surname of the man who said "the highest knowledge is to know that we surrounded by mystery"`。

The password is schweitzer.

附录

1 Caesar密码

```c
#include <math.h>
#include <stdio.h>
#include <string.h>

int main() {
  char input[128] = {0};
  scanf("%s", input);
  for (int i = 1; i < 26; i++) {
    for (int j = 0; input[j] != 0; j++) {
      input[j] = (input[j] + 1 - 'A') % 26 + 'A';
    }
    printf("key: +%d\t", i);
    for (int j = 0; j < strlen(input); j++) {
      printf("%c", input[j] - ('A' - 'a'));
    }
    printf("\n");
  }
  return 0;
```

```
}
```

2 Vigenere密码

```
#include <math.h>
#include <stdio.h>
#include <string.h>

int main() {
  char input[512] = {0};
  char output[512] = {0};
  char d[512][512] = {0};
  char secret[512] = {0};
  // char s[26] = {'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'};
  double freq[26] = {0.08167, 0.01492, 0.02782, 0.04253, 0.12705, 0.02228,
                     0.02015, 0.06094, 0.06996, 0.00153, 0.00772, 0.04025,
                     0.02406, 0.06749, 0.07507, 0.01929, 0.0009,  0.05987,
                     0.06327, 0.09056, 0.02758, 0.00978, 0.02360, 0.0015,
                     0.01974, 0.00074};
  int l = 1;  // the number of substrings

  scanf("%s", input);
  while (1) {
    double coincidence[512] = {0};
    int n, m;
    double index = 0;
    for (int i = 0; i < 512; i++) {
      for (int j = 0; j < 512; j++) {
        d[i][j] = 0;
      }
    }
    // guess the length of the secret key
    for (int i = 0; i < strlen(input); i++) {  // get substring
      n = i % l;
      m = i / l;
      d[n][m] = input[i];
    }
    for (int i = 0; i < l; i++) {
      int alphabet[26] = {0};
      int len = strlen(d[i]);
      for (int j = 0; j < len; j++) {
        alphabet[d[i][j] - 'a']++;
```

```
    }
    for (int j = 0; j < 26; j++) {
      coincidence[i] += alphabet[j] * (alphabet[j] - 1);
    }
    coincidence[i] /= len * (len - 1);
    printf("%lf\t", coincidence[i]);
    index += coincidence[i];
  }
  index = index / l;
  printf("%lf\n", index);
  if (index > 0.06 && index < 0.07) {
    break;
  }
  l++;
}
printf("The length of the secret key: %d\n", l);
// decrypt the secret key
for (int i = 0; i < l; i++) {
  int len = (int)strlen(d[i]);// length of the substring
  int offset = 0;
  double maxchi = 0;  // max chi
  int alphabet[26] = {0};
  for (int j = 0; j < len; j++) {
    alphabet[d[i][j] - 'a']++;
  }
  for (int j = 0; j < 26; j++) {  //j: offset
    double chi = 0;
    for (int k = 0; k < 26; k++) {
      chi += (alphabet[(k + j) % 26] / (double)len) * freq[k];
    }
    if (chi > maxchi) {
      maxchi = chi;
      offset = j;
    }
  }
  secret[i] = offset + 'a';
  printf("\n");
}
for (int i = 0; i < (int)strlen(input); i++) {
  output[i] = (d[i % l][i / l] - secret[i % l] + 26) % 26 + 'a';
}
printf("The secret key: %s\n", secret);
printf("The output: %s\n", output);
return 0;
```

```
}
```

3 Unknown密码

```
#include <stdio.h>
#include <string.h>

int main() {//统计字母频率
  char input[128] = {0};
  int alphabet[26] = {0};
  scanf("%s", input);
  for (int i = 0; i < (int)strlen(input); i++) {
    alphabet[input[i] - 'A']++;
  }
  for (int i = 0; i < 26; i++) {
    printf("%c: %lf\t", 'a' + i, (double)alphabet[i] / strlen(input));
  }
  printf("\n");
  return 0;
}
```