密码学实验报告 3

张天辰 17377321

2019 年 3 月 20 日

1 仿射密码

1.1 简介

仿射密码就是基于 $c \equiv a * m + b \pmod{26}$ 的加密方法，其中要求 $\gcd(a, 26) = 1$。

1.2 算法实现

Algorithm 1 仿射密码

输入: 密钥 a, b，明文 m（加密）或密文 c（解密）

输出: 密文 c（加密）或由密钥得到的明文 m'

```
function AffineEncrypt(message, a, b)
    cipher ← []
    for each m ∈ message do
        cipher.append((a * int(m) + b) mod 26)
    end for
    return cipher → string
end function
function AffineDecrypt(cipher, a, b)
    message ← []
    for each c ∈ cipher do
        message.append((int(c) − b)*reverse(a) mod 26)
    end for
    return message → string
end function
```

1.3 测试样例

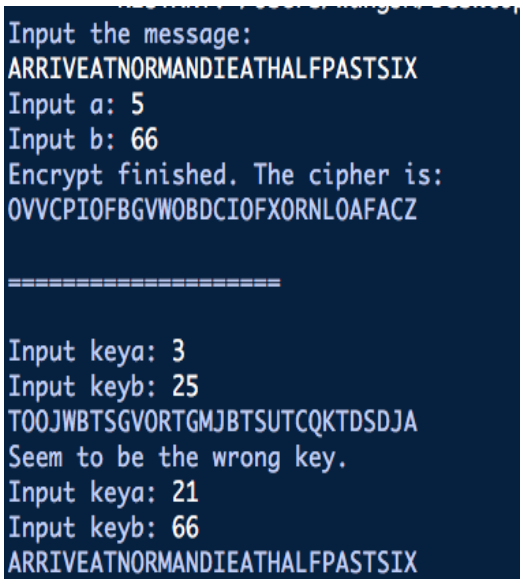

图 1: Affine

2 Vigenere 密码

2.1 简介

此加密方案即将明文和密钥（循环使用）的对应字母相加再模 26 得到的文字作为密文。

Algorithm 2 Vigenere 密码

输入: 密钥 key，明文 $message$（加密）或密文 $cipher$（解密）

输出: 密文 $cipher$（加密）或明文 $message$（解密）

function VigenereEncrypt($message, key$)
  $cipher \leftarrow []$
  $lenK \leftarrow len(key)$
  for each $i \in [0, len(message))$ **do**
    $temp \leftarrow (message[i] + key[i\%lenK]) \mod 26$
    $cipher.append(temp)$
  end for
  return $cipher \rightarrow string$
end function
function VigenereDecrypt($cipher, key$)
  $message \leftarrow []$
  $lenK \leftarrow len(key)$
  for each $i \in [0, len(message))$ **do**
    $temp \leftarrow cipher[i] - key[i\%lenK]$

15: $message.append(temp)$
16: **end for**
17: **return** $message \rightarrow str$
18: **end function**

2.2 测试样例

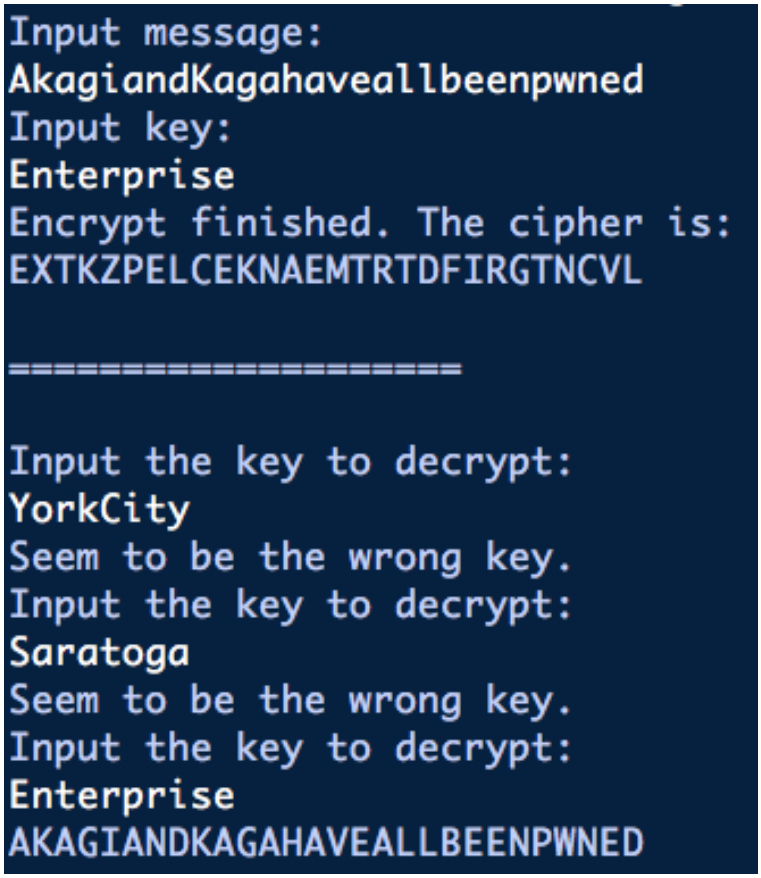

图 2: *Virgenere*

3 Vernam 密码

3.1 算法简介

该密码方案将明文和密钥转换为二进制码，然后按位异或（密钥循环）得到密文。解密方法与加密相同。特别地，为了实现中文加密，如果输入语言为中文则每个字符占 16 位，如果为英文则为 8 位。

3.2 算法实现

Algorithm 3 Vernam 密码

输入: 密钥 *cipher*，明文 *message*（加密）或密文 *cipher*（解密）
输出: 明文 *message*（解密）或密文 *cipher*（加密）

```
function VernamEncrypt(message, key)
    cipher ← []
    tempm ← bin(message)
    tempk ← bin(key)
    lenK ← len(key)
    for each i ∈ [0, tempm) do
        cipher.append(tempm ⊕ tempk → str)
    end for
    return cipher
end function
function VernamDecrypt(cipher, key)
    message ← []
    tempc ← bin(cipher)
    tempk ← bin(key)
    lenK ← len(key)
    for each i ∈ [0, tempc) do
        cipher.append(tempc ⊕ tempk → str)
    end for
    return message
end function
```

3.3 测试样例

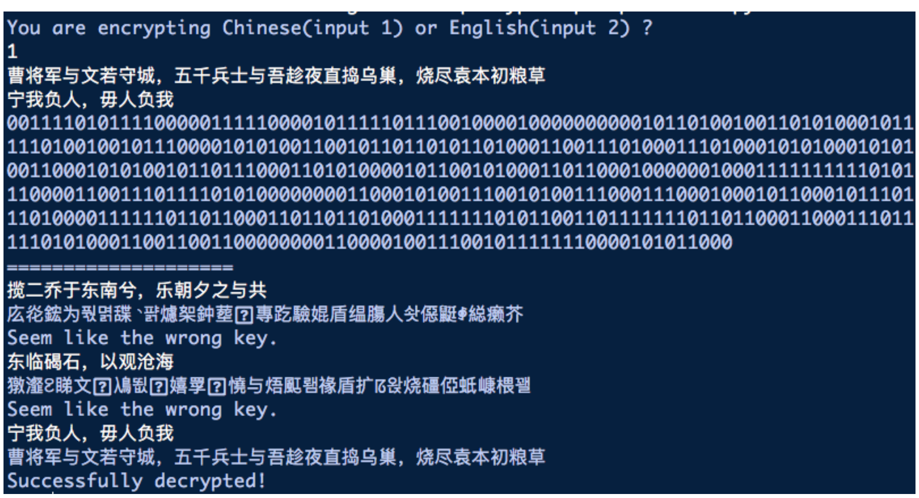

图 3: $Vernam$

4 单表代替密码

4.1 简介

利用词频分析可以攻击单表代替密码。通过给不同的字母频率分级并进行枚举猜测，可以实现这种攻击。首先将高频的 e，t 字母单独处理，在高频的两个密文字母中轮换；再处理 a、e、i、o 四个中频字母，在四个中频密文字母中轮换；最后按照概率顺序处理其余的所有字母。

输出顺序也与概率相关。因为 e 概率大于 t，a、e、i、o 概率依次递减，所以按照全排列生成的顺序就是理论上概率从大到小的顺序。本算法按照这个顺序进行输出。

4.2 算法实现

Algorithm 4 Hill 密码及其破解

输入: 密文 *cipher*

输出: 明文 *message*，密钥 *key*

```
dict_high ← e,t 两个高频字母
dict_medium ←a, o, i, n 四个中频字母
dict_low ← 其余字母
function STATISTICS(cipher)
    statis ←
    将每个字母写入字典的 key，出现频率作为 value
    statis 按照 value 排序
end function
function CRACK(cipher)
    probab ←STATISTICS(cipher)
    keyd ←
    for probab[0, 1] 分别对应 e, t do
        对应写入 keyd
        for probab[2, 5] 分别对应 a，o，i，n 全排列 do
            对应写入 keyd
            其余按概率排序分别写入 keyd
            对应字典解密，输出
        end for
    end for
end function
```

4.3 测试样例

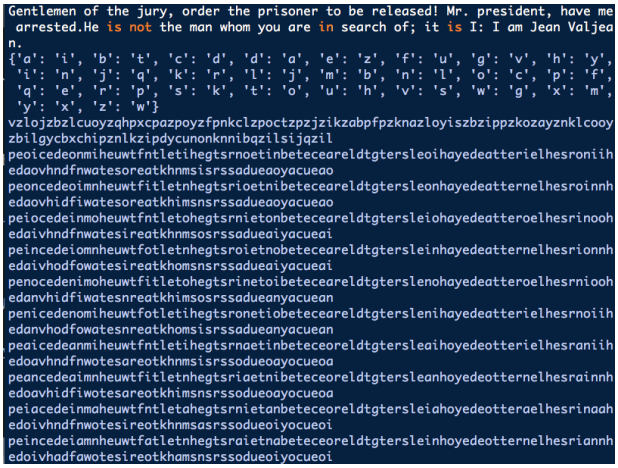

图 4: *SubstitutionCipher*

5 Hill 密码

5.1 简介

此加密方案用矩阵作为密钥，对明文消息利用矩阵乘法加密。解密时利用乘矩阵逆元解密。同时，如果获得足够多明密文对，可以对其进行攻击。设明文列向量排成矩阵 M，其加密结果排成矩阵 C，密钥为 K，则有：

$$KM = C \quad K = CM^{-1}$$

可以实现对密钥的攻击。

为了提高程序运行效率，本算法使用 python 自带函数求解矩阵行列式。如果自己写代码，在矩阵阶数较高（如 256）时，其计算量将是惊人的。此外，为了实现支持中文加密，本方案放弃了 mod 26 的矩阵，而是选择了在 GF_2 上构造矩阵。这样的优点是保证了所有运算能够在域上完成。

矩阵求逆算法采用高斯消元法，复杂度为 $O(n^3)$。

5.2 算法实现

Algorithm 5 Hill 密码及其破解

输入: 密钥 *cipher*，明文 *message*（加密）或密文 *cipher*（解密）

输出: 明文 *message*（解密）或密文 *cipher*（加密）

1: **function** REVERSE(a n)
2: **for** each $k \in [0, n)$ **do**
3: 找到 $a[k][k]$ 及其右下角子式中最大元，并与通过行列变换与 $a[k][k]$ 交换
4: 交换的行列保存在 $rmax[], cmax[]$
5: **for** each $j \in [0, n)$ **do**
6: **if** $j \neq k$ **then** $a[k][j] \leftarrow a[k][j] * a[k][k]$

```
            end if
        end for
        for each i ∈ [0, n) do
            if i ≠ k then
                for each j ∈ [0, n) do
                    if j ≠ k then a[i][j] ← a[i][j] ⊕ (a[i][k] ∗ a[k][j])
                    end if
                end for
            end if
        end for
        for each i ∈ [0, n) do
            if i ≠ k then a[i][k] ← a[i][k] ∗ a[k][k]
            end if
        end for
    end for
    将之前每次循环开始做的行、列变换按照后到先出顺序做其逆变换
    return a[][]
end function
function HillEncrypt(key, message, size)
    intm ← message → bit
    cipher ← []
    for i = 0; i < len(intm); i+ = size do cipher.append(key·intm[i : i + size])
    end for
    return cipher → str
end function
function HillDecrypt(cipher, key, size)
    intc ← cipher → bit
    message ← []
    key_1 →reverse(key, size)
    tempk ← bin(key)
    lenK ← len(key)
    for i = 0; i < len(intc); i+ = size do
        message.append(key_1·intc[i : i + size])
    end for
    return message → str
end function
function HillPwn(message, cipher, size)
    intm ← message → bit
```

45: $intc \leftarrow cipher \rightarrow bit$

46: **repeat**$message$ 中取 $size^2$bit 组成矩阵 m_mat $cipher$ 中取 $size^2$bit 组成矩阵 c_mat

47: **until** DET$(m_mat) \neq 0$

48: **return** c_matm_mat

49: **end function**

5.3 测试样例

图 5: *Hill*

6 感想

作业量好大，时间好短。

单表代替密码果然没有那么容易破译，这样破解出来的东西完全没有可读性。如果想要破解这种密码，还是需要语言学家参与进行推断。这样简单的推断还是不够的，需要大量的语言学字典才可以。

希望这门课能变好。