密码仙人设计指南

Github@WangTingZheng

仿射密码原理

仿射密码的数学定义

明文空间 M：a 到 z，26 个字母
密文空间 C：a 到 z，26 个字母
密文：K={k_1,k_2}，其中，k_1,k_2 为整数，且满足：

$$k_1 \geq 0; k_1 \leq 25$$

$$k_1 k_1^{-1} = 1 mod(26) \tag{1}$$

$$\gcd(k_1, 26) = 1 \tag{2}$$

翻译成大白话就是，k_1和k_2是介于 1 到 26 之间的正整数，而且k_1与 26 的公约数是 1
对于任意的：

$$m \in M, c \in C, k = (k_1, k_2) \in K$$

加密变换：

$$c = E(m) = (k_1 m + k_2) mod(26) \tag{3}$$

解密变换：

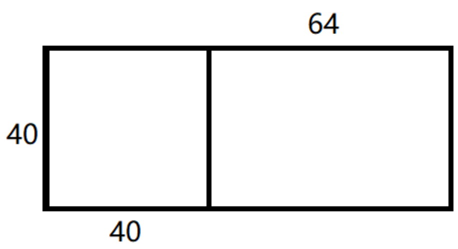
$$m = D(c) = [k_1^{-1}(c - k_2)] mod(26) \tag{4}$$

k_1合法性校验

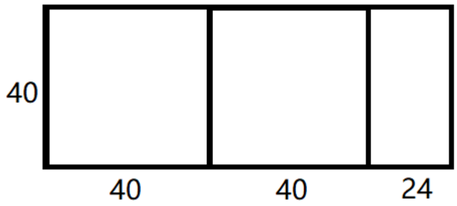
如何确定输入的 k 的值符合密文的条件？大小和是否是整数就不说了，关键在最后一个条件：

$$\gcd(k_1, 26) = 1$$

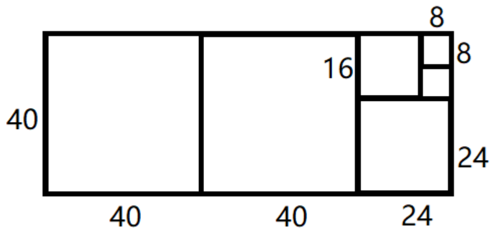
我们要证明k_1与 26 的公约数为 1，如果不满足的话，明文字母和密文字母将无法一一对应。那我们该如何证明呢？有的人说这还不简单！求出它们两的公约数不就可以了吗？可问题就出现在这个求公约数上，现在我们将使用一个叫辗转相除法的算法来求两个数的公约数。
想象一下，你有一个 40X104 大小长方体，你现在需要用一个未知边长的正方形来填满整个长方形，不能有多，也不能有少的，这该是怎么样的一个过程呢？
首先，你可以放入一个最大的正方形，它的大小是 40X40：



那么整个长方形还剩下一个 64X40 的长方形，你继续填充一个 40X40 的长方形：



一直填上去，直到最后，你会发现，当你填入一个 8X8 的正方形的时候，你刚好就能将整个长方形填满！



你惊奇地发现，整个 8X8 的正方形也能填满它上一级的 16X16 的方块，也能填满 24X24 的方块，也能填满 40X40 的方块，也就是说，8X8 的方块完全可以填满整个 40X104 的方块！到此，你用画图的方法找到了 40 和 104 的最大公约数，也就是：

$$\gcd(40,106) = 1$$

而且任意的一组数都能通过这个方法来求出它们的公约数，但是我们不能每一次都画图啊，有没有一种数学方法，能够算出来最大公约数？实际上是有的，正是我们上文提到的辗转相除法。还是以$\gcd(40,104)$为例，我们进行一下运算：

$$106 = 40 * 2 + 24$$
$$40 = 24 * 1 + 16$$
$$24 = 16 * 1 + 8$$
$$16 = 8 * 2$$

这么样？很熟悉吧，上面的过程正是对填长方形的数学模拟，第一个式子的意思是我们把 104 的边长分为两段 40 的，剩下一段 24 的，我们就把长方形分成两个 40X40 的和一个 40X24 的，第二个式子表示我们把剩下的 40X24 的长方形进行分割，因为剩下的边长一定比 40 小，使用我们那 40 开刀，40 可以分割成一段 24 的和一段 16 的，以此类推，最后我们至于把长方形分割成没有剩余的情况了，最后的余数毫无疑问就是最大公约数，这个也能通过上文的画图法看出来。

所以当我们知道k_1的值时，便能通过辗转相除法来计算出它和 26 的最小公约数，以次来判断k_1是否满足条件，到此，k_1的合法性判断就完成了。现在，我们假设我们的密钥空间是：

$$K = \{k_1, k_2\} = \{9,2\} \tag{5}$$

我们来计算一下 9 和 26 的最小公约数：

$$26 = 9 * 2 + 8$$
$$9 = 8 * 1 + 1$$
$$8 = 8 * 1$$

我们可以看到，最后一个余数是 1，所以$\gcd(9,26) = 1$，满足$\gcd(k_1, 26) = 1$的条件。而且k_1, k_2也都是处在 0 到 25 之间的整数，所以这个密钥空间是符合条件的。

加密与解密

接下来，我们就可以进行加密了，我们先得把明文从字母转化为可以计算的整数，比如说 a 转化为 1，我们以 china 这个明文为例，这个过程就是：

$$china \rightarrow \{2,7,8,13,0\} \tag{5}$$

这样我们就可以根据加密公式，也就是公式 3 来对明文进行加密，因为次数公式中的 k_1, k_2, m我们都知道了，这个过程就是：

$$c = k_1 m + k_2$$
$$\{2,7,8,13,0\} \cdot 9 + \{2,2,2,2,2\} = \{20,13,22,15,2\}$$

这样我们就得到了密文空间，为了让它们，我们先对它对 26 取 26，把大于 25 的数转化为 0 到 25 的数，在把它转化成字母：

$$\{20,13,22,15,2\} mod(26) = \{20,13,22,15,2\} = \{u,n,w,p,c\}$$

这样我们就得到了最后的密文！

好了，现在我们会加密了，我们应当试试解密了！让我们瞧瞧公式：

$$m = [k_1^{-1}(c - k_2)] mod(26)$$

我们发现了一件很不幸的事：我们不知道式子中k_1^{-1}的值，这意味着我们不能愉快地代公式了，我们必须先把k_1^{-1}求出来，我们知道，k_1^{-1}满足下面的条件：

$$k_1 k_1^{-1} = 1 mod(26)$$

在这个公式中，我们知道k_1的值，但是不知道k_1^{-1}的值，我们可以使用一个叫辗转相除法的算法来求这个值。

首先我们要把问题一般化：

已知：

$$a, b, x, m \in Z^+$$

有如下等式：

$$ax \equiv b mod(m) \tag{4}$$

求出整数 x 的值。

我们来求解这个问题。我们没看过这个式子的这种形式，所以我们应当把它变换一下：

$$ax = ym + b \qquad y \in Z$$

再稍微转化一下，移一下项：

$$ax - ym = b$$
$$ax + y(-m) = b \tag{5}$$

那么我们如果求这个 x 呢？我们可以使用一个叫拓展欧几里得算法，它其实就是前面的辗转相除（欧几里得算法）倒过来。

拓展欧几里得算法是这样表述的：

对于：

$$a, b, m \in Z$$

等式：

$$ax + by = m$$

有整数解的充要条件是：

$$m = k \cdot \gcd(a, b) \qquad k \in Z$$

用自然语言描述就是，如果 m 是 a 和 b 的最大公约数的整数倍，则$ax + by = m$一定有整数解，反过来讲，如果$ax + by = m$有整数解，则 m 是 a 和 b 最大公约数的整数倍。

那么，如果有整数解，那么这个整数解是多少呢？这是我们关心的，我们可以再使用一下前面的辗转相除法，我们以$40x+104=8$为例：

$$104=40*2+24 \tag{6-1}$$

$$40=24*1+16 \tag{6-2}$$

$$24=16+8 \tag{6-3}$$

$$16=8*2 \tag{6-4}$$

根据上面的式子，我们可以从公式 6-3 得到：

$$8=24-16 \tag{7-1}$$

我们也可以从公式 6-2 得到：

$$16=40-24 \tag{7-2}$$

我们可以把公式 7-2 代入公式 7-1：

$$8=24-(40-24)=24*2-40 \tag{7-3}$$

我们可以从公式 6-1 得出：

$$24=104-40*2 \tag{7-4}$$

我们再把 7-4 代入 7-3：

$$8=(104-40*2)*2-40=104*2-40*5 \tag{7-5}$$

我们最终得到公式 7-5，但是有有什么用呢？我们观察原始式子：

$$40x+104y=8$$

还看不出来？再转化一下：

$$8=104*2+40*(-5)$$

$$8=104y+40x$$

我相信你能看出来：

$$y=2$$

$$x=-5$$

这样我们就能求出这个式子的解了，但还一个问题：一定只有一个解吗？答案是不一定的，我们只求出了一个解，这个解我们称作x_0，实际上，如果$\gcd(a,b)=d$，$\frac{a}{d}=n$的话，所有解满足：

$$x_n=(x_0+n\cdot t_n)mod(m) \qquad t_n=0,1,2,3,4\ldots d$$

把已知量代入这个方程，就能求出方程的所有的解。

现在，我们拿以前的例子来算算k_1^{-1}的值吧！

已知：

$$k_1\cdot k_1^{-1}\equiv 1mod(26)$$

也就是：

$$9x=26\cdot y+1$$

$$9x+26(-y)=1$$

我们来进行辗转相除法：

$$26=2*9+8$$

$$9=1*8+1$$

$$8=8*1$$

我们在把9,16,26以线性的形式联系起来：

$$1=9-8$$

$$8=26-2*9$$

$$1=9-(26-2*9)=9*3-26$$

$$9*3+26(-1)=1$$

我们再把原始式子拿过来进行对比

$$9x+26(-y)=1$$

$$9*3+26(-1)=1$$

可得:

$$x=3, y=1$$

所以$x_0=3$

我们再求其它值:

$$\gcd(9,26)=1$$

$$\frac{m}{d}=\frac{1}{1}=1$$

所以只有一个解$X=\{x_0\}=\{3\}$，所以$k_1^{-1}=3$，接下来我们可以解密了:

$$m=D(c)=[k_1^{-1}(c-k_2)]mod(26)$$

$$m=[3\cdot(\{20,13,22,15,2\}-\{2,2,2,2,2\}]=\{54,33,60,39,0\}$$

我们再进行$mod(26)$运算:

$$m=\{2,7,8,13,0\}$$

再化成字母得形式:

$$\{2,7,8,13,0\}\rightarrow\{c,h,i,n,a\}$$

刚好等于我们的明文！bingo!

递归求逆元

在上文中，我们已经介绍了仿射密码的原理，当然这还不够，我们还需要在计算机上实现仿射密码的加密与解密，我们观察加密与解密的公式，我们可以发现，除了逆元以外，其它步骤都比较好计算，所以如何在计算机上求得一个数的逆元，就是我们重点需要考虑的问题了。这里提供一个递归求逆元的方法。

我们设

$$AB = 1mod(N)$$

则我们认为，B 是 A 的逆元，我们可以把上面这个式子变换一下：

$$AB = xN + 1$$

然后我们使用辗转相除法：

$$N = A \cdot a_0 + r_0$$
$$A = r_0 \cdot a_1 + r_1$$
$$r_0 = r_1 \cdot a_2 + r_2$$
$$r_1 = r_2 \cdot a_3 + r_3$$
$$\dots\dots\dots\dots\dots\dots\dots.$$
$$r_{n-2} = r_{n-1} \cdot a_n + r_n$$
$$r_{n-1} = r_n \cdot a_{n+1} + r_{n+1}$$

那么，我们设

$$b_{-1} = 1;$$
$$b_0 = a_n$$
$$b_i = a_{n-i} \cdot b_{i-1} + b_{i-2}$$

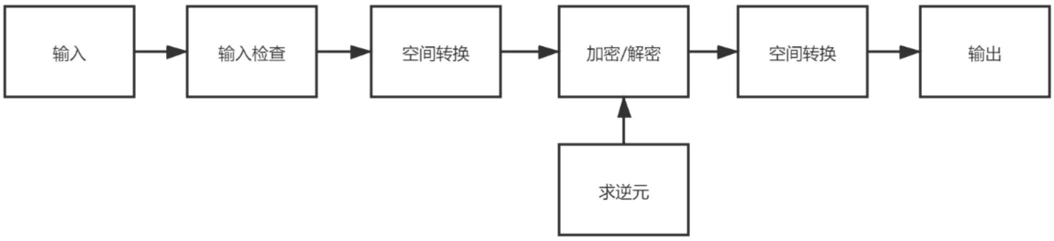
如果 n 为奇数，则逆元$B = b_n$

如果 n 为偶数，则逆元$B = N - b_n$

有了这个定理，我们就可以使用递归的方式来求逆元了，首先我们需要用递归辗转相除法保存所有的a，然后再来一个递归求出b_i

程序设计

通过上面的原理设计，我们可以把程序划分为下面几个模块，输入模块负责处理用户输入的字符串，把它转化为程序能够识别的数据结构，这一块的大概思路是使用正则表达式进行匹配，后期还支持个性化扩展；输入检查模块的主要工作是检查输入的明文/密文空间、密钥空间是否要求，如果不合格，及时终止；空间转换模块的功能是把字符型的明文/密文空间转换为可以被运算的整型明文/密文空间或者把转换好的整型明文/密文转换为可阅读的字符型明文/密文空间；加密/解密模块的主要功能是把明文/密文空间转换为密文/明文空间，对于解密，还有有一个求逆元的过程；输出模块的主要功能是输出已经加密或者解密好的空间，供用户阅读。



输入模块

未写

输入检查模块

在这个模块中，我们要实现对明文空间，密文空间，密钥空间进行检查。

空间检查

对于输入的明文空间或者密文空间，要满足以下要求才能进行变换：

- 列表的每一个元素都是字符
- 列表的每一个元素都只有一个字符
- 列表的每一个字符都是小写字母

我们可以遍历列表中的每一个元素，通过检查每一个元素是否符合要求来检查整个列表，如果有一个元素不满足一个条件，函数可以立刻执行 **return False** 来退出检查函数并且提示不满足条件。具体的函数代码如下：

```
def check_list(inf_list):
    """
    检查用户输入的明文或密文空间是否合法
    :param inf_list: 字符 list，输入的密文或明文空间，每一个元素存一个字母，字母只能是小写的，从左往右
```

```python
    :return: 如果合法，返回 True，否则返回 False
    """
    for i in inf_list:
        if type(i) is not str:  # 如果有元素不是字符型的，不合法
            return False
        elif len(i) != 1:  # 如果一个元素不止有一个字符的，不合法
            return False
        elif ord(i) < 97 or ord(i) > 122:  # 如果字符不是小写字母，不合法
            return False
    return True
```

密钥空间检查

密钥空间的检查就简单多了，当输入密钥空间中的一个时，需要判断其是否是整数就可以了，当然，完全存在这个值不在 0 到 25 之间的情况，如果出现了这种情况，我们只需要整除 26 取余就可以了，效果是一样的。

```python
def check_key(a):
    """
    检查密钥空间是否合法
    :param a: 整型变量，要检查的密钥空间的其中一个
    :return: 如果合法，返回 mod(26)后的密钥（整型变量），否则返回 False
    """
    if type(a) is not int:  # 如果密钥不是整型，不合法
        return False
    return a % 26  # 如果合法，则返回 mod（26)的密钥
```

k_1的检验

对于密钥空间里的k_1，我们还需要判断一下是否符合$\gcd(k_1, 26) = 1$，所以我们必须能算出两个数的最大公约数，我们可以使用递归的方式：

```python
def gcd(a, b):
    """
    求两个整数的最大公约数，输入的 a 必须大于 b
    :param a: 整型变量，其中一个数
    :param b: 整型变量，另一个数
    :return: 整型变量，a 和 b 的最大公约数
    """
    if a % b == 0:
        return b
    return gcd(b, a % b)
```

但是这个函数对输入参数有大小关系的要求，不是很方便，我们再进行一层封装：

```python
def return_gcd(a, b):
    """
    返回 a 与 b 的最大公约数，不需要考虑 a 和 b 的大小关系
    :param a: 整型变量，其中一个数
    :param b: 整型变量，另一个数
    :return: 整型变量，a 和 b 的最大公约数
    """
    if a > b:
        return gcd(a, b)
    else:
        return gcd(b, a)
```

这样我们就能求出 a 与 b 的最大公约数了

合并

有了上面两个函数，我们只需要将它们合并起来就可以了，主要的检查顺序是：

- 检查密钥空间的密钥是否是整数
- 检查k_1与 26 的最大公约数是否为 1
- 检查输入的空间列表是否符合仿射密码的要求

具体的代码如下：

```python
def check_key_all(convert_list, k1, k2):
    """
    检查所有输入值函数
    :param convert_list: 字符 list，需要检查的明文/密文空间
    :param k1: 整型变量，密钥空间中的 k1
    :param k2: 整型变量，密钥空间中的 k2
    :return: 如果全合法，则返回由 mod（26)后 k1,k2 组成的整型 list，否则，返回 False
    """
    k1 = check_key(k1)
    k2 = check_key(k2)
    if k1 is False and k2 is False:
        return False
    elif convert.gcd(k1, 26) != 1:
        return False
    elif check_list(convert_list) is False:
        return False
    return k1, k2
```

空间转换模块

我们要实现两个功能，把小写字母转化为整数，对应关系是 a-0, z-25，我们可以使用 ASCII 码进行转化，把小写字母的 ASCII 减去 97 就是对应的整数，我们还需要把整数转化为小写

字母，我们可以吧整数加 97，再转化为小写字母，具体的代码如下：

```
def char_to_int(char_list):
    """
    把字符密文/明文空间 list 转化为整数 list，规则是 a 对应 0，z 对应 25
    :param char_list: 要转化的字符密文/明文空间 list
    :return: 转化好的整型密文/明文空间 list
    """
    res = []
    for i in char_list:
        res.append(ord(i) - 97)
    return res


def int_to_char(char_list):
    """
    把整型密文/明文空间 list 转化为字符 list，规则是 0 对应 a，25 对应 z
    :param char_list: 要转化的整型密文/明文空间 list
    :return: 转化好的字符密文/明文空间 list
    """
    res = []
    for i in char_list:
        res.append(chr(i + 97))
    return res
```

加密/解密模块

求逆元

由上面的思路可知，我们需要拿到辗转相除法中每一个式子中的除数，我们只需要写一个递归，在每一次递归中把除数加入到一个列表，最后返回列表就可以了：

```
def return_a_in_affine(a, b, temp_list):
    """
    使用辗转相除法，拿到 a,b 列的式子中的每一个除数 c 并转化为一个 list，接到 temp_list 后面, a 必须大于 b
    公式是 ax=dmod(b)  或者  a = b*c + d
    :param a: 整数，总数中的已知量
    :param b: 整数，被除数
    :param temp_list: 一个任意的 list
    :return: list，一个原来的 list 加上所有 c 的集合
    """
    c = a // b
    d = a % b
    if d == 0:
```

```
        return temp_list
    else:
        temp_list.append(c)
    a = b
    b = d
    return return_a_in_affine(a, b, temp_list)
```

但是上面的函数对 a 和 b 的大小关系有要求，所以我们还得写一个函数进行进一步的封装：

```
def return_list(a, b):
    """
    使用辗转相除法，拿到 a,b 列的式子中的每一个除数 c 并转化为一个 list，接到 temp_list 后面, a 不用必须大于 b
    公式是 ax=dmod(b)  或者  a = b*c + d
    :param a:  整数，总数中的已知量
    :param b:  整数，被除数
    :return: list，所有 c 的集合
    """
    if a > b:
        return return_a_in_affine(a, b, [])
    else:
        return_a_in_affine(b, a, [])
```

接下来，我们可以拿我们得到的除数的列表进行处理，再使用递归来计算 b，进而算出逆元，代码如下：

```
def return_inverse_element(a, b):
    """
    返回逆元，即 aa-1=1mod(b),返回 a-1
    :param a:  整型，所要求逆元的整数
    :param b:  整型，模数，被除数
    :return:  整型，逆元
    """
    list_a = return_list(a, b)
    b_s_1 = 1
    b_s_2 = list_a[len(list_a) - 1]

    def return_inverse_element_loop(b_1, b_2, n):
        if n != len(list_a):
            b_t_2 = list_a[len(list_a) - n - 1] * b_2 + b_1
        else:
            if n % 2 != 0:
                return max(a, b) - b_2
            return b_2
        b_t_1 = b_2
        n = n + 1
        return return_inverse_element_loop(b_t_1, b_t_2, n)
    return return_inverse_element_loop(b_s_1, b_s_2, 1)
```

加密与解密

有了整数的密文/明文空间和密钥，我们就可以进行加密或解密操作了，对于解密，还一个求逆元的动作，具体的代码如下，都是套公式：

```
def encoding(char_list, k1, k2):
    """
    加密函数，不带字母与整数之间的转化
    :param char_list: 整型 list，明文空间，由字母明文空间转化而来
    :param k1: 整型，密钥中的 k1
    :param k2: 整型，密钥中的 k2
    :return: 整型 list，加密之后的整数密文空间，可转化为字母密文空间
    """
    res = []
    for i in char_list:
        res.append((i * k1 + k2) % 26)    # 根据仿射密码加密公式编写
    return res


def decoding(char_list, k1, k2):
    """
    解密函数，不带字母与整数之间的转化
    :param char_list: 整型 list，密文空间，由字母密文空间转化而来
    :param k1: 整型，密钥中的 k1
    :param k2: 整型，密钥中的 k2
    :return: 整型 list，解密之后的整数明文空间，可转化为字母明文空间
    """
    res = []
    k_1 = affine.return_inverse_element(k1, 26)
    for i in char_list:
        res.append((k_1 * (i - k2)) % 26)    # 根据仿射密码解密公式编写
    return res
```

合并测试

我们去除输入模块，用简单的 print 函数代替输出模块，然后把上面整个过程串联起来：

```
def encoding_or_decoding(convert_list, k1, k2, mode):
    """
        完整的加密函数，包含了检查、转化功能
        :param convert_list: 字符 list，明文空间，list 每个元素是一个字母，从左往右
        :param k1: 整型，密钥中的 k1
        :param k2: 整型，密钥中的 k2
        :param mode: 字符串，模式选择，可以传入"encoding"和"decoding"，如果有其它传入，
终止变换并报错
```

```python
    :return: 字符 list，加密之后的密文空间，list 每一个元素是一个字母，从左往右
    """
    k_list = check.check_key_all(convert_list, k1, k2)  # 检查明/密文空间、密钥是否合法
    if k_list is False:
        return False
    k1 = k_list[0]
    k2 = k_list[1]
    if k1 is False and k2 is False:
        return False

    convert__int_list = convert.char_to_int(convert_list)  # 把字母明文空间转化为整数明文空间
    if mode == "encoding":
        res_int_list = encoding(convert__int_list, k1, k2)  # 加密整数明文空间
    elif mode == "decoding":
        res_int_list = decoding(convert__int_list, k1, k2)  # 加密整数明文空间
    else:
        print("mode 参数错误，请输入\"encoding\"或者\"\decoding\"")
        return False
    res_char_list = convert.int_to_char(res_int_list)  # 把加密完的整数密文空间转化为字符密文空间
    return res_char_list
```

文言文编程

未写

原理证明

未写