默克尔-赫尔曼背包密码

Merkle-Hellman Knapsack Cipher

刘卓

1 欧几里得算法-伯利坎普

欧几里得算法 (Euclidean Algorithm), 也称为辗转相除法。是计算两个给定整数 A 和 B 的最大公因数 d 的过程。其中 $A > B$。

伯利坎普对欧几里得算法略微进行修改，可以计算得到 B 的逆 mod A(inverse of B mod A)。也称伯利坎普算法 (Berlekamp Algorithm).

算法步骤：

1. 设 $r_{-2} = A, r_{-1} = B, p_{-2} = 0, p_{-1} = 1, q_{-2} = 1, q_{-1} = 0$;
2. 对于 $k = 0, 1, 2, ...$, 逐个计算;

$$
\begin{aligned}
a_k &= r_{k-2} \div r_{k-1}(\text{取整}) \\
r_{k-2} &= a_k r_{k-1} + r_k \\
p_k &= a_k p_{k-1} + p_{k-2} \\
q_k &= a_k q_{k-1} + q_{k-2}
\end{aligned}
$$

3. 当 $r_n = 0$, 停止计算;

然后 $r_{n-1} = gcd(A, B)$。此外，在这种情况下 $gcd(A, B) = 1$，并且

$$B \cdot (-1)^n p_{n-1} = 1 + A \cdot (-1)^n q_{n-1}$$

如果 $gcd(A, B) \neq 1$, 则 B^{-1} 不存在。

例 1

计算 115 的逆并 mod 12659.

解:

k	r	a	p	q
-2(初始)	12659(A)		0(初始)	1(初始)
-1(初始)	115(B)		1(初始)	0(初始)
0	9	110	110	1
1	7	12	1321	12
2	2	1	1431	13
3	1	3	5614	51
4(停止计算)	0	2	12659(A)	115(B)

$B^{-1} = (-1)^n P_{n-1} = (-1)^4 5614 = 5614$ □

2 子集和问题

子集和问题 (Subset-Sum Problem)。给定一串递增数列 $a_1 < a_2 < \cdots < a_n$ 和一个目标数字 M。问递增数列是否存在某个非空子集，使得子集内中的数字和为 M。即

$$x_1a_1 + x_2a_2 + \cdots + x_na_n = M$$

其中 x_i 为 0 或 1，x_i 不全为 0

定义：**超级递增序列 (a super-increasing sequence)** 是序列中每一个数都大于它之前所有数的总和。即：

$$a_k > \sum_{i=0}^{k-1} a_i$$

注意，有时候存在多组解。

例 2

数列 $[3, 5, 11, 23, 51], M = 67$，尝试解决 SSP 问题。

解：

$$67 = 51 + 11 + 5 = a_2 + a_3 + a_5$$

$$x = [0, 1, 1, 0, 1]$$

即存在一组序列 x 使该问题可以被解决。 □

例 3

数列 $[13, 18, 35, 72, 155, 301, 595], M = 1003$，尝试解决 SSP 问题。

解：

$$\begin{aligned} 1003 &= 595 + 408 \\ &= 595 + 301 + 107 \\ &= 595 + 301 + 72 + 35 \\ &= a_3 + a_4 + a_6 + a_7 \\ x &= [0, 0, 1, 1, 0, 1, 1] \end{aligned}$$

即存在一组序列 x 使该问题可以被解决。 □

3 默克尔-赫尔曼背包密码

加密过程：

1. Bob作为收信人 (receiver)
   - 选择一个超级递增序列 (a super-increasing sequence) a = $(a_1, a_2, ..., a_n)$
   - 选择一个质数 $p > \sum_{i=1}^{n} a_i$
   - 选择一个加密因子 A，A 必须满足 $2 \leq A \leq p-1$
   - 保留为秘密，不公开
2. Bob计算出 b_i 序列，$b_i = A \cdot a_i \bmod p$，然后把序列发送给Alice作为发信人 (sender)
3. Alice收到Bob发送的序列后，需要利用这串序列加密明文，假设明文为二进制信息 $x_1x_2x_3...x_n$，由 n 个 0 或 1 组成。x_i 为 ASCII 二进制的所在位置，计算

$$C = x_1b_1 + x_2b_2 + ... + x_nb_n$$

   然后将 C 发送给Bob。注意：如果字符串长度少于 n，则补齐；长度大于 n，则截取分段发送。
4. Bob收到Alice发送的信息后，开始解密。先计算 $M = A^{-1}C \bmod p$, 然后解决 SSP 问题

$$x_1a_1 + x_2a_2 + \cdots + x_na_n = M$$

   计算得出序列 X，即可得知二进制信息 $x_1x_2...x_n$，再通过 ASCII 表格翻译成字符。

例 4

Bob选择了一个序列

$$(a_1, a_2, \ldots, a_8) = (2, 5, 9, 22, 47, 99, 203, 409)$$

和质数 $p = 997$ 和加密因子 $A = 60$.

1. 则序列 b 是

$$b = 60 \times (2, 5, 9, 22, 47, 99, 203, 409) \bmod 997 = (120, 300, 540, 323, 826, 955, 216, 612)$$

   Alice收到序列 b 后，加密信息"b",b 的 ASCII 二进制代码为：01100010.
2. 则 C 为：

$$C = 0 \cdot b_1 + 1 \cdot b_2 + \cdots + 0 \cdot b_8 = 1056$$

3. Bob收到 C 后，尝试解密。使用 Python 代码计算 A^{-1}，方法为：

$$pow(A,\ -1,\ p)$$

$$A^{-1} = \ pow(60,\ -1,\ 997) = 781$$

4. 计算 $M = A^{-1} \cdot C \bmod p$

$$M = 781 \cdot 1056 \bmod 997 = 217$$

5. 解决 SSP 问题

$$217 = 5 + 9 + 203 = a_2 + a_3 + a_7$$

$$X = (0, 1, 1, 0, 0, 0, 1, 0)$$

6. 根据二进制代码转化为字符**'b'**

例 5

```
Plaintext = 'AGREE' #明文

a = [2,5,9,22,47,99,203,409] #Bob设置超级递增数列
p = 997 #Bob设置质数
A = 60 #Bob设置加密因子

b  = [i*A%p for i in a] #求b数列，发送给Alice

A_inv = pow(A,-1,p) #Bob求A的逆

def subset_sum(t,s,x,n,M,a,X):
    """
    子集和问题求解，返回空集合或者序列X

    Parameters
    ----------
    t : int
        递归深度.
    s : int
        子集和.
    x : dict
        判断列表，状态为1或0.
    n : int
        序列长度.
    M : int
        序列长度.
    a : list
        集合b，使得x_1b_1+x_2b_2+...+x_nb_n=M.
    X : list:
        空list，储存结果

    Returns
    -------
    X: list
    0 或 1.
    """
    if t == n:
        if s == M:
            for i in range(0,n):
                if x[i] != 0:
                    X += [1] #取b_i
                else:
                    X += [0] #不取b_i

    else:
        s = s+a[t]
```

```
            x[t] = a[t]
            subset_sum(t+1,s,x,n,M,a,X)
            s = s-a[t]        #回溯之前还原
            x[t] = 0
            subset_sum(t+1,s,x,n,M,a,X)

        return X


for i in Plaintext:
    """Alice 加密过程"""
    c_int = ord(i)
    c_bin = bin(c_int) #将字符转化为二进制
    c = 0

    if len(c_bin[2:]) != 8:
        c_bin = '0'*(8-len(c_bin[2:])) + c_bin[2:] #二进制转为8位二进制，与a长度相同
    else:
        c_bin = c_bin[2:]

    for j in range(8):
        try:
            # print(c_bin[j],b[j],j)
            c += int(c_bin[j])*b[j] #计算c，然后发送给Bob
        except:
            pass
    # print(c)
     """Bob 解密过程"""
    M = A_inv * c%p
    # print(M)

    t = 0           #递归深度
    s = 0           #子集和
    x = {}          #判断列表，状态为1或0
    n = len(a)
    X = []

    X = subset_sum(t,s,x,n,M,a,X) #解决子集和问题
    # print(X)

    bin_X = ''
    for k in X:
        bin_X += str(k)
    print(chr(int(bin_X,2))) #二进制转化ASCII字符
```

输出：AGREE