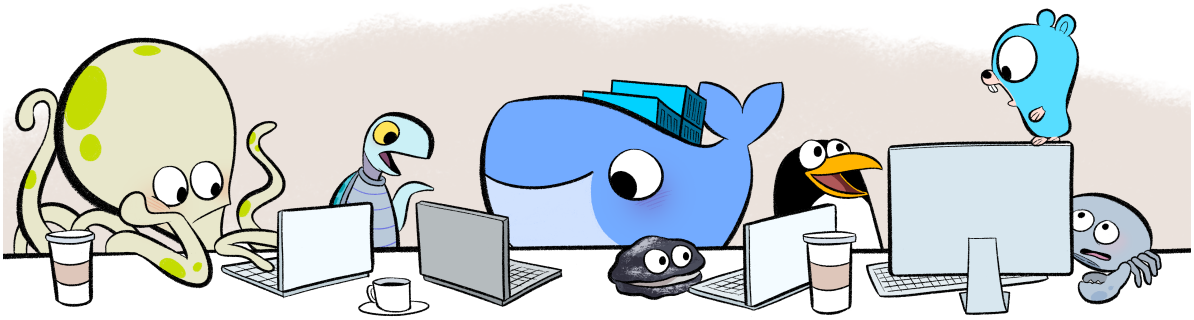
Shamir密钥分享算法

2022/5/4

http://zhongmy.xyz/2020/05/26/Python%E5%AE%9E%E7%8E%B0Shamir%E5%AF%86%E9%92%A5%E5%88%86%E4%BA%AB%E7%AE%97%E6%B3%95%E7%9A%84%E8%A7%A3%E5%AF%86/



0x00 | 前言

大二下网络信息安全上机作业，正好我也爱嗯造web，就搞了

本文作者：forimoc.com | forimoc.me | forimoc2021@gmail.com | 2097517935@qq.com

0x01 | 原理

Shamir的密钥分享算法的功能是将秘密分为`n`个子秘密，任意`t`个子秘密都可以恢复原秘密，而任意小于`t`个子秘密都无法得到原秘密的任何信息。数学原理用到的是线性代数中的秩，即`t`为秩而`n`为总的等式数，取得任意线性不相关的等式子集合后都能解出线性方程组（好久没看线性代数，描述的有点稀烂）

具体原理上面的链接感觉讲的比较详细了，然后下面放的代码也有很多注释，就不细说了。一个需要注意的点是计算拉格朗日多项式时不能直接除分母，而是乘上分母的乘法逆元，这一功能可以通过扩展欧几里得算法实现

0x02 | 代码实现

只写了python的代码实现，一开始想写的是golang的代码，但是go强类型还是驾驭不了，特别是`big.Int`这个大数类不太会用，以及各种类型转换也玩不溜，还是PHP、Python这种弱类型语言比较适合本脚本小子

```python
# -*- coding: utf-8 -*-
# @Time : 2022/5/4 9:10
# @File : main.py
# @Software : PyCharm
import random
import functools
#  定义有限域的模数
PRIME = 2 ** 127 - 1
#  生成随机整数的函数
RINT_FUNC = functools.partial(random.SystemRandom().randint, 0)


#  计算多项式在 x 处的值
#  参数: 多项式系数数组 poly , 整数 x , 有限域的模数 p
def value_at(poly, x, p):
    value = 0
    for i in range(len(poly)):
        value += poly[i] * x ** i
    return value % p


#  生成真正的秘密(poly[0])与分享的密钥点集(points[i][j])
#  参数: 阈值(门限) t , 分享的密钥数 n , 有限域的模数 p
def make_random_shares(t, n, p=PRIME):
    if t > n:
        raise ValueError("门限不能大于密钥总数!")
    #  随机生成多项式的系数 0 ~ t-1 共 t 个, poly[0] 为真正的秘密
    poly = [RINT_FUNC(p - 1) for i in range(t)]
    #  生成 n 个密钥
    points = [(i, value_at(poly, i, p)) for i in range(1, n + 1)]
    return poly[0], points


#  扩展欧几里得算法, 用来计算乘法逆元
#  参数: ax+by=r 中的 a , b
def egcd(a, b):
    if b == 0:
        return a, 1, 0
    r, x, y = egcd(b, a % b)
    return r, y, x - a // b * y


#  拉格朗日插值算法, 通过密钥还原真正的秘密
#  参数: 可用的密钥点集 selected_points , 有限域模数 p
def lagrange_interpolate(selected_points, p):
    s = 0
    for i in range(len(selected_points)):
        up = 1
        down = 1
        for j in range(len(selected_points)):
            if i != j:
                up *= -selected_points[j][0]
                down *= selected_points[i][0] - selected_points[j][0]
```

```
        #  使用乘法逆元代替直接相除
        item = (up * egcd(down, p)[1]) % p
        s += item * selected_points[i][1]
    return s % p


if __name__ == '__main__':
    secret, points = make_random_shares(t=5, n=8)
    print("真正的秘密: ", secret)
    print("分享的密钥: ", points)
    print("使用前5个密钥进行解密获得的秘密: ", lagrange_interpolate(points[:5],
PRIME))
    print("\n")
    print("真正的秘密: ", secret)
    print("分享的密钥: ", points)
    print("使用前4个密钥进行解密获得的秘密: ", lagrange_interpolate(points[:4],
PRIME))
    print("显然，使用不足门限值个数的密钥无法还原真实秘密!", secret, "!=",
lagrange_interpolate(points[:4], PRIME))
```

0x03 | Web应用实现

构建了一个Flask的Web应用来实现Shamir算法，这里展示的只是最基础的只有逻辑的版本（引用了jquery库），前端美化以及各自前端框架就不搞了。需要参考具体文件的话https://github.com/FORIMOC请便

Demo展示网址：Shamir Demo

> 为什么用flask：因为python的web服务中flask是我认知中最轻的了，如果golang的话我会用gin

app.py

```
from flask import Flask, render_template, request
import json
import random
import functools
PRIME = 2 ** 31 - 1
RINT_FUNC = functools.partial(random.SystemRandom().randint, 0)
app = Flask(__name__)


@app.route('/')
def index():
    return render_template("index.html")


@app.route('/generate', methods=['POST'])
def generator():
    def value_at(poly, x, p):
        value = 0
        for i in range(len(poly)):
            value += poly[i] * x ** i
        return value % p

    def make_random_shares(t, n, p=PRIME):
```

```
        if t > n:
            raise ValueError("门限不能大于密钥总数!")
        poly = [RINT_FUNC(p - 1) for i in range(t)]
        points = [(i, value_at(poly, i, p)) for i in range(1, n + 1)]
        return poly[0], points

    if request.method == 'POST':
        t = int(request.values.get('t'))
        n = int(request.values.get('n'))
        if t > n:
            resp = {
                'code': 422,
                'msg': "门限不能大于密钥总数!"
            }
            return json.dumps(resp)
        secret, points = make_random_shares(t, n, PRIME)
        resp = {
            'code': 200,
            'secret': secret,
            'points': points
        }
        return json.dumps(resp)


@app.route('/decrypt', methods=['POST'])
def decryptor():
    def egcd(a, b):
        if b == 0:
            return a, 1, 0
        r, x, y = egcd(b, a % b)
        return r, y, x - a // b * y

    def lagrange_interpolate(selected_points, p):
        s = 0
        for i in range(len(selected_points)):
            up = 1
            down = 1
            for j in range(len(selected_points)):
                if i != j:
                    up *= -selected_points[j][0]
                    down *= selected_points[i][0] - selected_points[j][0]
            item = (up * egcd(down, p)[1]) % p
            s += item * selected_points[i][1]
        return s % p

    if request.method == 'POST':
        data = request.get_data()
        data = json.loads(data)
        points = data.get('points')
        for point in points:
            point[1] = int(point[1])
        secret = lagrange_interpolate(points, PRIME)
        resp = {
            'code': 200,
            "decrypted_secret": secret
        }
        return json.dumps(resp)
```

```
if __name__ == '__main__':
    app.run(host="0.0.0.0")
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Shamir Demo</title>
    <script src="/static/js/jquery-3.6.0.js"></script>
</head>
<body>
<header>
    <h1>Hello Shamir!</h1>
</header>
<div class="generate">
    <div class="params">
        t: <input id="t" type="number">
        n: <input id="n" type="number">
        <input id="generator" type="submit" value="生成秘密和密钥">
    </div>
    <div class="secret-box">
        <div id="secret"></div>
        <div id="decrypted_secret"></div>
        <div id="points"></div>
    </div>
</div>

<div class="decrypt">
    <input id="decryptor" type="submit" value="解密">
</div>

</body>
<script>
    // 生成秘密和密钥
    $('#generator').click(function () {
        var t = document.getElementById('t').value
        var n = document.getElementById('n').value
        var data = {
            t: t,
            n: n,
        }
        $.post('/generate', data, function (result) {
            result = JSON.parse(result)
            $('#secret').html('')
            $('#decrypted_secret').html('')
            $('#secret').append('<p>secret: '+result.secret+'</p>')
            $('#points').html('')
            for (var i=0;i<result.points.length;i++){
                 $('#points').append('<p class="point"><input
type="checkbox"/>'+result.points[i][1]+'</p>')
            }
        })
    })
```

```
    // 解密
    $('#decryptor').click(function () {
        var selected_points = []
        var i=0
        $('.point').each(function (index) {
            if ($(this).children('input[type=checkbox]').prop('checked')){
                selected_points.push([])
                selected_points[i].push(index+1)
                selected_points[i].push($(this).html().replace(/<[^>]+>/g, ""))
                i++
            }
        })
        var data = {
            points: selected_points
        }
        $.ajax({
            type: 'post',
            url: '/decrypt',
            data: JSON.stringify(data),
            dataType: 'json',
            contentType: 'application/json',
            success: function (result){
                $('#decrypted_secret').html('')
                $('#decrypted_secret').append('<p>decrypted_secret:
'+result.decrypted_secret+'</p>')
            }
        })
    })
</script>
</html>
```

效果展示

`t=3 n=5`，选中两个子密钥，解出来的秘密和真实秘密不相同，解密失败

选中三个子密钥，解出来的秘密和真实秘密相同，解密成功

Hello Shamir!

t: 3 n: 5 生成秘密和密钥

secret: 1562401466

decrypted_secret: 1562401466

- [x] 100174364
- [ ] 1052740057
- [x] 125131251
- [x] 1612315240
- [ ] 1219324730

解密