moduhash

zer0pts ctf/crypto/226pts/16solves

Sikesibian
2023/8/3

题目描述

```python
CC = ComplexField(256)
for _ in range(100):
    n = randint(32, 64)
    h1 = to_hash(gen_random_hash(n))

    zi = CC.random_element()
    print(f"zi  : {zi}")
    print(f"h1(zi): {hash(zi, h1)}")

    h2 = input("your hash> ")

    if not hash_eq(h1, h2, CC):
        print("your hash is incorrect")
        exit()

print(flag)
```

在SageMath中，CC = ComplexField(256)是将一个复数域(CC)定义为256位精度的操作。ComplexField(256)是一个复数域的构造函数，它将返回一个具有给定位数精度的复数域对象。

具体而言，ComplexField(256)定义了一个复数域，其中实部和虚部都是256位精度的浮点数。这意味着在进行浮点数计算时，保留了256位的精度，以提高计算的准确性。使用这样的复数域对象，可以执行高精度计算，特别是涉及到复数运算的情况。

例如，可以使用CC来进行复数运算，例如加法、减法、乘法、除法以及其他常见的数学运算。使用256位精度的复数域可以提供更准确的计算结果，并且适用于需要高精度计算的特定应用领域，如数值分析、科学计算、加密等。

每一轮可以获得的信息：

1. 哈希**函数**h_1每轮**随机生成**，且**长度不固定**
2. **原像**z_i每轮**随机生成**
3. 能够**知道**每轮**原像**z_i和**像**$hash(z_i, h_1)$
4. 每轮需要我们**快速生成**与h_1**等效**的h_2

```python
def hash(z, h):
    res = z
    for s in h:
        if s == "S":
            res = S(res)
        elif s == "T":
            res = T(res)
        else:
            exit()
    return res

def hash_eq(h1, h2, CC):
    for _ in range(100):
        zr = CC.random_element()
        h1zr = hash(zr, h1)
        h2zr = hash(zr, h2)
        if abs(h1zr - h2zr) > 1e-15:
            return False
    return True
```

第一个函数中z为一个复数，hash函数对其做h变换。h由S和T两种变换组合而成，即形如：

$$h = S^{i_1} T^{j_1} S^{i_2} T^{j_2} \cdots$$

第二个函数是一个检验哈希的函数，要求检验100次，每次复数都是随机生成的，要求我们给出的哈希函数h_2与题目生成的h_1效果相当（差值模长小于1e-15）

期待：

1. 最好能够还原出h_1
2. 检查是否有类似收敛的特性
3. 能否构造标准型

```python
import os

flag = os.environb.get(b"FLAG", b"dummmmy{test_test_test}")

def S(z):
    return -1/z

def T(z):
    return z + 1

def gen_random_hash(n):
    r = bytes([getrandbits(8) for _ in range(0, n)])
    return r

def to_hash(st):
    res = ""
    for s in st:
        sts = bin(s)[2:].zfill(8)
        for x in sts:
            if x == "0":
                res += "S"
            else:
                res += "T"
    return res
```

***S*变换：**

$$\mathcal{S}\colon \mathbb{C} \longrightarrow \mathbb{C}$$
$$z \longmapsto -\frac{1}{z}$$

***T*变换：**

$$\mathcal{T}\colon \mathbb{C} \longrightarrow \mathbb{C}$$
$$z \longmapsto z+1$$

初步观察：

显然 $\mathcal{S} \circ \mathcal{S} = id$

因此每一个h均会形如：

$$h = \mathcal{T}^{j_1}\mathcal{S}\mathcal{T}^{j_2}\cdots$$

实践发现，貌似存在一些收敛现象

题目背景

模群

模群 Modular Group

$$SL_2(\mathbb{Z}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \middle| a, b, c, d \in \mathbb{Z}, ad - bc = 1 \right\}$$

模群中的逆元

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \longrightarrow \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

模群的群作用

$$\gamma \in SL_2(\mathbb{Z}) \qquad SL_2(\mathbb{Z}) \times H \to H \qquad H = \{z \in \mathbb{C} | \mathrm{Im}(z) > 0\}$$

上半复平面

$$\gamma(z) = \frac{az + b}{cz + d}$$

注：主同余子群、同余子群等也是重要的相关概念，这里便不多加介绍

参考：GTM 228: *A First Course in Modular Forms*, Fred Diamond, Jerry Shurman

模群的有限生成性

$(ST)^3 = -E \qquad S^2 = -E$

$\mathcal{T}(z) = z + 1 \quad \mathcal{S}(z) = -\dfrac{1}{z}$

$SL_2(\mathbb{Z})$由$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$与$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$在矩�乘法下生成

S变换矩阵 → $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} -c & -d \\ a & b \end{pmatrix}$

T变换矩阵 → $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a + nc & b + nd \\ c & d \end{pmatrix}$

参考：GTM 228: *A First Course in Modular Forms*, Fred Diamond, Jerry Shurman

模群的群作用

$$SL_2(\mathbb{Z}) \times H \to \quad H$$

$$\gamma(z) \quad = \frac{az+b}{cz+d}$$

相似

$$z_1 \sim z_2 \Leftrightarrow z_2 = \gamma(z_1)$$

轨道

$$\mathcal{O}_x = \{\gamma(z) | \gamma \in SL_2(\mathbb{Z})\}$$

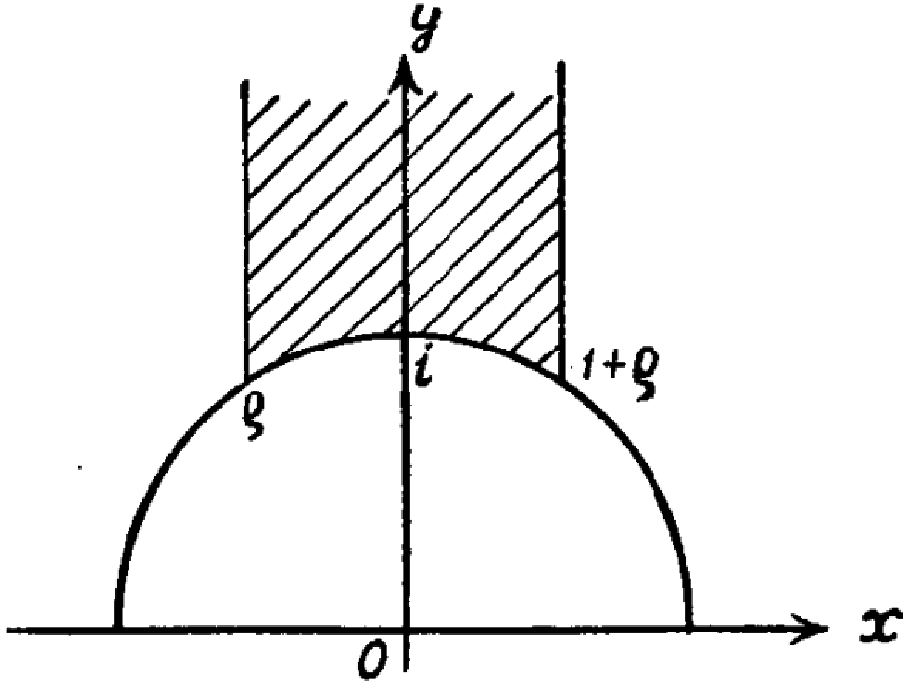
参考：GTM 228: *A First Course in Modular Forms*, Fred Diamond, Jerry Shurman

基域

$SL_2(\mathbb{Z})$的基域/基本域

$$D=\left\{z\in H\,\middle|\,|\mathrm{Re}(z)|\le\frac{1}{2},|z|\ge 1\right\}$$

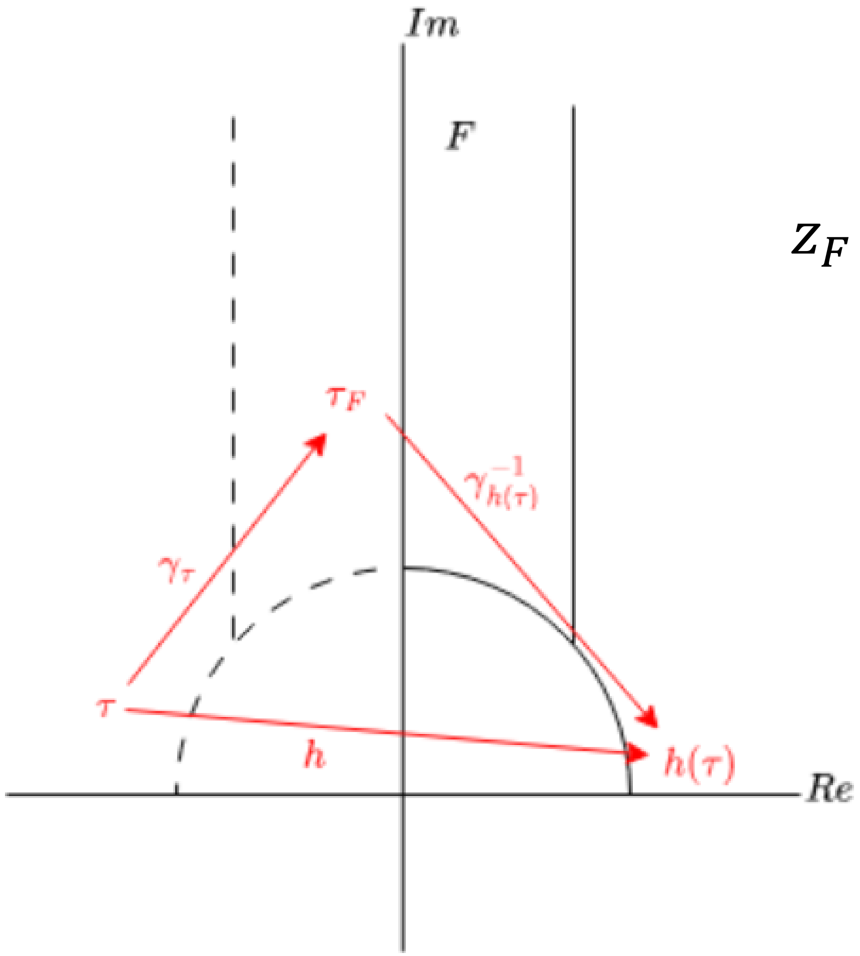
H：上半复平面



- 上半复平面所有点都能放到基域D中（不漏性）
- 若基域中的两个不相等的复数在同一轨道上，那么它们一定在基域的边界上（不重性）

题目解答

方法一



$$z_F = \gamma_z(z)$$

$$z_F = \gamma_{h(z)}\big(h(z)\big) \Rightarrow h(z) = \gamma^{-1}_{h(z)}(z_F)$$

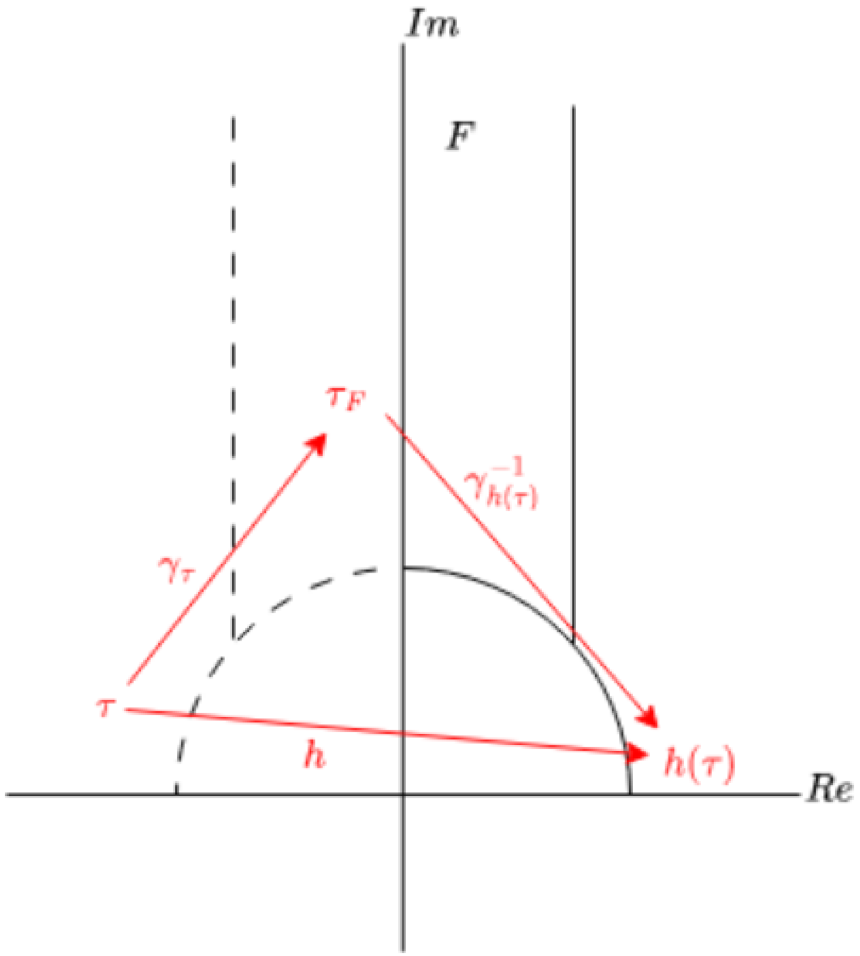
$$\rightarrow \quad h_2 = \gamma^{-1}_{h(z)} \circ \gamma_z$$

两个要素：

1. 找到两个变换的逆
2. 如何将平面上的复数规约到基域中去

参考：zer0pts ctf 2023 writeup (English) (mitsu1119.github.io)

方法一



两个要素:

1. 找到两个变换的逆
2. 如何将平面上的复数规约到基域中去

$$(ST)^3 = -E \quad \Rightarrow \quad \mathcal{T}^{-1} = \mathcal{STSTS} = \mathcal{U}$$

$$S^2 = -E \quad \Rightarrow \quad \mathcal{S}^{-1} = \mathcal{S}$$

注：左边是矩阵，右边是变换，注意矩阵和变换的差异

由于基域的宽度就是1，所以我们在任何时候都可以通过平移变换将实部变得满足要求，这时我们就只需要关心虚部能否移动到基域中。如果虚部不能落入，那么总可以将其挪到单位圆内，然后利用S变换将其反射出去，再平移即可。

参考：zer0pts ctf 2023 writeup (English) (mitsu1119.github.io)

方法一

```python
def S(z):
    return -1/z

def T(z):
    return z + 1

def U(z):
    return z - 1
```

```python
def in_fundamental(z):
    if -0.5 <= z.real() and z.real() <= 0.5 and abs(z) >= 1:
        return True
    return False

def gen_to_fundamental_hash(z):
    res = ""
    while True:
        while True:
            if z.real() > 0.5:
                res += "U"
                z = U(z)
            elif z.real() < -0.5:
                res += "T"
                z = T(z)
            else:
                break
        if abs(z) < 1:
            res += "S"
            z = S(z)
        if in_fundamental(z):
            break
    res = res.replace("U", "STSTS").replace("SS", "").replace("TSTSTS", "")
    return res
```

```python
def hash_inv(st):
    res = ""
    for s in st:
        if s == "S":
            res = "S" + res
        else:
            res = "U" + res
    res = res.replace("U", "STSTS").replace("SS", "").replace("TSTSTS", "")
    return res
```

参考：zer0pts ctf 2023 writeup (English) (mitsu1119.github.io)

方法二

$$SL_2(\mathbb{Z}) \times H \to \qquad H$$

$$\gamma(z) = \frac{az+b}{cz+d}$$

$$\downarrow$$

$$h_1(z) = \frac{az+b}{cz+d}$$

$$\Rightarrow \quad (cz+d)h_1(z) = az+b$$

$$\Rightarrow \quad c{\color{green}h_1(z)}z + d{\color{cyan}h_1(z)} = az + b\cdot 1$$

$$\Rightarrow \quad \boldsymbol{c}{\color{green}\boldsymbol{g}} + \boldsymbol{dh} = \boldsymbol{az} + \boldsymbol{b}\cdot{\color{red}\mathbf{1}}$$

$$\begin{cases} \boldsymbol{c}{\color{green}\boldsymbol{g}_{\mathrm{Re}}} + \boldsymbol{d}{\color{cyan}\boldsymbol{h}_{\mathrm{Re}}} = \boldsymbol{a}{\color{orange}\boldsymbol{z}_{\mathrm{Re}}} + \boldsymbol{b}\cdot{\color{red}\mathbf{1}} \\ \boldsymbol{c}{\color{green}\boldsymbol{g}_{\mathrm{Im}}} + \boldsymbol{d}{\color{cyan}\boldsymbol{h}_{\mathrm{Im}}} = \boldsymbol{a}{\color{orange}\boldsymbol{z}_{\mathrm{Im}}} + \boldsymbol{b}\cdot{\color{red}\mathbf{1}} \end{cases}$$

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccccc} \phantom{2^{256}z_{\mathrm{Re}}} & \phantom{2^{256}z_{\mathrm{Im}}} & a & b & c & d \end{array} \\ \begin{pmatrix} 2^{256}{\color{orange}z_{\mathrm{Re}}} & 2^{256}{\color{orange}z_{\mathrm{Im}}} & 1 & 0 & 0 & 0 \\ 2^{256}\cdot{\color{red}1} & {\color{red}0} & 0 & 1 & 0 & 0 \\ -2^{256}{\color{green}g_{\mathrm{Re}}} & -2^{256}{\color{green}g_{\mathrm{Im}}} & 0 & 0 & 1 & 0 \\ -2^{256}{\color{cyan}h_{\mathrm{Re}}} & -2^{256}{\color{cyan}h_{\mathrm{Im}}} & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

参考：zer0pts CTF 2023 Writeups｜廢文集中區 (maple3142.net)

方法二

S变换矩阵

$$S\cdot\begin{pmatrix}a & b\\ c & d\end{pmatrix}=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\cdot\begin{pmatrix}a & b\\ c & d\end{pmatrix}=\begin{pmatrix}-c & -d\\ a & b\end{pmatrix}$$

T变换矩阵

$$T^n\cdot\begin{pmatrix}a & b\\ c & d\end{pmatrix}=\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}^n\cdot\begin{pmatrix}a & b\\ c & d\end{pmatrix}=\begin{pmatrix}a+nc & b+nd\\ c & d\end{pmatrix}$$

$$\rightarrow\quad ST^{-n}\cdot\begin{pmatrix}a & b\\ c & d\end{pmatrix}=\begin{pmatrix}-c & -d\\ a-nc & b-nd\end{pmatrix}$$

$$\Big\downarrow \text{类似于辗转相除法}$$

$$ST^{-n_1}ST^{-n_2}\cdots\begin{pmatrix}a & b\\ c & d\end{pmatrix}=\begin{pmatrix}\pm1 & n'\\ 0 & \pm1\end{pmatrix}=S^{1\mp1}T^{n'}$$

$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\sim\mathcal{T}^{n_k}\mathcal{S}\cdots\mathcal{T}^{n_2}\mathcal{S}\mathcal{T}^{n_1}\mathcal{S}\mathcal{S}^{\alpha}\mathcal{T}^{n'}\qquad \alpha=0,2$$

方法二

```
def find(z, h):
    g = z * h
    L = matrix(QQ,
        [
            [ z.real(),  z.imag(), 1, 0, 0, 0],
            [        1,         0, 0, 1, 0, 0],
            [-g.real(), -g.imag(), 0, 0, 1, 0],
            [-h.real(), -h.imag(), 0, 0, 0, 1],
        ])
    L[:, :2] *= 2**256
    L = L.LLL()
    return L[0][2:]
```

```
def positive_mod(x, y):
    r = x % y
    if r < 0:
        r -= y
    return r
```

```
def decompose(a, b, c, d):
    M = matrix(ZZ, [[a, b], [c, d]])
    S = matrix(ZZ, [[0, -1], [1, 0]])
    T = matrix(ZZ, [[1, 1], [0, 1]])

    M0 = M
    res = []
    res_s = ""
    for _ in range(200):
        if M[0, 0] == 0 or M[1, 0] == 0: break
        while abs(M[1, 0]) > abs(M[0, 0]):
            M = ~S * M
            res.append(S)
            res_s += "S"
        while sign(M[0, 0]) != sign(M[1, 0]):
            M = ~S * M
            res.append(S)
            res_s += "S"
        a, c = M[0, 0], M[1, 0]
        r = positive_mod(a, c)
        q = (a - r) // c
        M = T ^ (-q) * M
        res.append(T ^ q)
        res_s += "T" * int(q)

    assert prod(res) * M == M0
    return res, res_s[::-1]
```

参考：zer0pts CTF 2023 Writeups | 廢文集中區 (maple3142.net)

更多延伸内容……

推荐参考

- *A First Course in Modular Forms*, Fred Diamond, Jerry Shurman (GTM 228)
- 《数论导引》, 华罗庚
- 《模形式导引》, 潘承洞
- 《模形式初步》, 李文威

Thanks!