

利用生日悖论求解离散对数

生日悖论

离散对数 $y = g^x \bmod p$ (p 为素数)

若已知 y, g, p 求 x 就十分困难，最多需要穷举 p 次。

生日悖论是指，23人的群体中，两个或更多人拥有相同生日的概率大于0.5。

其含义是，看起来概率很小发生的事情，事实上它发生的概率却很大。

用生日悖论攻击离散对数的基本原理：

Step 2: 随机生成一个 $1 \sim p-1$ 之间的整数 a ，计算 $A = g^a \bmod p$ ，并将 a 和 A 填入 Tab A； \leftarrow

Step 3: 随机生成一个 $1 \sim p-1$ 之间的整数 b ，计算 $B = yg^b \bmod p$ ，并将 b 和 B 填入 Tab B； \leftarrow

Step 4: 重复 Step 2 和 Step 3，直到 $n = \lfloor \sqrt{p} \rfloor$ 次； \leftarrow

Step 5: 在表 Tab A 和 Tab B 中找重。如果找到 Tab A 中的某个 A 与 Tab B 中的某个 B 相等，则记下相对应的 a 和 b ；如果没有找到，则回到 Step1； \leftarrow

根据上述步骤，如果出现了 $A = B$ ，即 $g^a \bmod p = [(g^x \bmod p)g^b] \bmod p$

由模数乘法的性质 $[(g^x \bmod p)g^b] \bmod p = g^{b+x} \bmod p$

p 是素数，在循环群 Z_p^* 中，如果 g 是 Z_p^* 的发生元， g 的阶数就是 Z_p^* 的基数，也即 $\Phi(p) = p-1$

上述等式就可以推出 $a = b + x \pmod{p-1}$

$x = a - b \pmod{p-1}$

python代码

```
def discrete_logarithm(result, base, mod, max_iterations=0):
    if not all(isinstance(arg, int) for arg in [result, base, mod]):
        raise TypeError("All parameters must be integers. 参数必须是整数")

    success_probability = 1 - 0.5**max_iterations

    def create_tableA(base, mod):
        import random
        tableA = dict()
```

```

for counter in range(0, int(mod ** 0.5)):
    a = random.randint(0, mod - 1)
    ga = pow(base, a, mod)
    tableA[ga] = a # 键是ga,值是a,方便查询
return tableA # 用字典是因为字典是底层是用哈希表实现的, 查找速度非常快。

def create_tableB(result, base, mod):
    import random
    tableB = dict()
    for counter in range(0, int(mod ** 0.5)):
        b = random.randint(0, mod - 1)
        ygb = (result * pow(base, b, mod)) % mod
        tableB[ygb] = b
    return tableB

while max_iterations:
    max_iterations -= 1

    tableA = create_tableA(base, mod)
    tableB = create_tableB(result, base, mod)
    keysA = set(tableA.keys())
    keysB = set(tableB.keys())
    common_results = keysA & keysB
    if common_results:
        common_item = common_results.pop()
        a, b = tableA[common_item], tableB[common_item]
        return (a - b) % (mod - 1)
    else:
        continue

    raise TimeoutError(f"""
        Exceeded maximum iterations ({max_iterations}) without finding a solution.
        超过最大循环次数({max_iterations})次. 若确定参数无误. 尝试改变max_iterations的值? 或再试一次? 当前成功概率为{success_probability}
    """)

if __name__ == '__main__':
    y = 5125495
    g = 3
    p = 5767169

    print(discrete_logarithm(y, g, p))

```

Z257*下测试

3是 \mathbb{Z}_{257}^* 的生成元, 取 $g=3$

得到 $178 = 3^{150} \mod 257$

```
if __name__ == '__main__':
    y = 178
    g = 3
    p = 257
    attack(y, g, p)
```

程序运行结果为:

```
x = 150
```

用了两次就破解出来。

\mathbb{Z}_{40961}^* 下测试

3是 \mathbb{Z}_{40961}^* 的生成元，仍然取 $g=3$

提前算出 $35788 = 3^{20000} \bmod 40961$

```
if __name__ == '__main__':
    y = 35788
    g = 3
    p = 40961
    attack(y, g, p)
```

程序运行结果为:

```
20000
```

在 $\mathbb{Z}_{104857601}^*$ 下测试

3是 $\mathbb{Z}_{104857601}^*$ 的一个生成元，提前算出

$66132073 = 3^{12345678} \bmod 104857601$

```
if __name__ == '__main__':
    y = 66132073
    g = 3
    p = 104857601
```

结果:

```
12345678
```

换Z5767169*

3是 $\mathbb{Z}_{5767169}^*$ 的生成元，提前算出

$$5125495 = 3^{1234567} \bmod 5767169$$

```
if __name__ == '__main__':  
    y = 5125495  
    g = 3  
    p = 5767169  
    attack(y, g, p)
```

结果：

1234567