Project 6: GooglePasswordCheckup 验证
参考论文 https://eprint.iacr.org/2019/723.pdf 的 section 3.1 ，也即 Figure 2 中展示的协议，尝试实现该协议，（编程语言不限）。

一. 协议流程和数学原理

协议目标

让用户能够安全地检查自己的密码是否存在于服务器的泄露密码库中

同时满足：

1.服务器无法知道用户具体查询了哪个密码。
2.用户无法获取服务器存储的所有泄露密码。

协议流程

初始化阶段（Setup）

服务器拥有一个泄露密码数据库 DB = {p_1, p_2, ..., p_n}。

服务器选择一个伪随机函数(PRF)和一个密钥 k。

服务器预先计算所有 F(k, p)并存储（或使用布隆过滤器优化存储）。

查询阶段（Query）

(1) 用户 → 服务器：用户选择随机数 r，计算 x = H(pwd)·r·G。H 是哈希函数 G 是椭圆曲线基点），发送 x 给服务器。

(2) 服务器 → 用户：服务器计算 y = F(k, x)返回 y 给用户。

(3) 用户本地计算：用户解盲，计算 z = y^(1/r)得到 F(k, H(pwd))。

如果存在于服务器的 DB，则密码已泄露。

数学原理

椭圆曲线加密（ElGamal）
密钥生成：sk = 随机数, pk = sk * G（G 为基点）。
加密：c1 = r * G, c2 = M + r * pk（M 为明文映射的点）。
解密：M = c2 - sk * c1。

布隆过滤器
通过 k 个哈希函数将密码映射到 m 位的位数组中。

使用 bitarray 存储和检查位标志。

哈希到椭圆曲线点
密码通过 SHA-256 哈希后，映射到椭圆曲线上的点（用于加密位置信息）。

二．客户端思路

在基础的服务器端验证是否以外，增加了一个本地校验弱密码的环节。若用户输入的密码存于弱密码库中，会直接提示弱密码。

```
#客户端所存本地弱密码库
local_weak_passwords = {"123456", "654321"}


def run_protocol(password: str):  1 个用法
    if password in local_weak_passwords:
        print(f"密码 '{password}' 属于已知弱密码库")
        return
```

先对密码进行预处理，对输入的密码做 k 次哈希，映射到布隆过滤器的多个位置，然后使用 ElGamal 加密这些位置（隐藏真实查询内容）。

```
elgamal = ECCElGamal()
sk, pk = elgamal.keygen()
positions = k_hashes(password, k, m)
ciphertexts = [elgamal.encrypt(pk, pos) for pos in positions]
serialized = [[point_to_bytes(c1).hex(), point_to_bytes(c2).hex()] for (c1, c2) in ciphertext
```

然后与服务器交互，发送加密的位置数据给服务器。

```
response = requests.post( url: "http://localhost:5000/check_password", json={
    "positions": positions,
    "ciphertexts": serialized
})
```

接收服务器返回的加密结果，解密并判断泄露。通过解密后的位置与原始位置是否一致，判断密码是否已经泄露。

```
results = response.json()["masked_ciphertexts"]
decrypted = []
for i in range(k):
    c1 = bytes_to_point(results[i][0])
    c2 = bytes_to_point(results[i][1])
    m_dec = elgamal.decrypt(sk, c1, c2)
    decrypted.append(m_dec)
```

三．服务端思路

先初始化服务器，存储一个泄露密码库，并使用布隆过滤器存储这些密码的哈希位置，优化查询效率。

```
elgamal = ECCElGamal()
#服务器端已泄露密码
leaked_db = ["password", "qwerty"]
bf = BloomFilter(m, k)
for pwd in leaked_db:
    bf.add(pwd)
```

需要处理查询时，接收客户端发来的加密位置数据（ElGamal 加密的布隆过滤器位置），检查这些位置是否在布隆过滤器中命中。如果命中就返回加密的“真”结果，未命中就返回随机加密的噪声

```
def check_password():
    data = request.get_json()
    positions = data["positions"]
    ciphertexts = data["ciphertexts"]

    processed = []
    for i in range(len(positions)):
        pos = positions[i]
        c1 = bytes_to_point(ciphertexts[i][0])
        c2 = bytes_to_point(ciphertexts[i][1])
        if bf.bits[pos]:
            processed.append([point_to_bytes(c1).hex(), point_to_bytes(c2).hex()])
        else:
            mc1, mc2 = elgamal.homomorphic_mask(c1, c2)
            processed.append([point_to_bytes(mc1).hex(), point_to_bytes(mc2).hex()])

    return jsonify({"masked_ciphertexts": processed})
```

四．实验结果

预设密码如下：

```
#客户端所存本地弱密码库
local_weak_passwords = {"123456", "654321"}


def run_protocol(password: str):  1 个用法
    if password in local_weak_passwords:
        print(f"密码 '{password}' 属于已知弱密码库")
        return
```

```
elgamal = ECCElGamal()
#服务器端已泄露密码
leaked_db = ["password", "qwerty"]
bf = BloomFilter(m, k)
for pwd in leaked_db:
    bf.add(pwd)
```

实际运行结果：

```
C:\Users\涂佳桦\MobileFile\Scripts\python.exe "D:\sdusummer\project6
Password Checkup 客户端
请输入密码（exit退出）: 123456
密码 '123456' 属于已知弱密码库

请输入密码（exit退出）: password
Password: password
Original positions: [71, 78, 212]
Decrypted results:  [71, 78, 212]
密码已泄露

请输入密码（exit退出）: safety
Password: safety
Original positions: [46, 6, 221]
Decrypted results:  [None, None, None]
密码未泄露
```