Google Password Checkup 协议与实现

一、引言

随着互联网的普及，数据泄露事件频发，用户密码安全面临严峻挑战。Google Password Checkup 协议旨在解决一个关键问题：如何让用户检查自己的密码是否在已知数据泄露中，同时不向任何第三方暴露其密码信息。

本报告基于论文《On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality》(eprint.iacr.org/2019/723.pdf)中的 section 3.1 内容，详细阐述该隐私保护密码检查协议的设计原理及其 Python 实现。

二、协议设计

1、设计目标

该协议的核心目标是实现 "隐私保护的密码泄露检查"，具体包括：

- 用户能够验证自己的密码是否在泄露密码数据库中
- 服务器不能获取用户的密码或密码哈希
- 辅助服务器不能单独判断用户密码是否泄露
- 整个过程中不泄露用户的任何敏感信息

2、参与方

协议涉及三个主要参与方：

- 用户(User)：希望检查自己密码安全性的个体
- 服务器(Server)：存储泄露密码哈希集合的服务提供者
- 辅助服务器(Helper)：协助进行隐私计算的第三方服务

3、协议流程

协议流程遵循论文中 Figure 2 所示的步骤，具体如下：

用户准备阶段：

- 用户对自己的密码进行哈希处理，得到 h_p
- 生成随机数 r
- 计算 $a=h_p$ XOR r 和 b=r
- 将 a 发送给服务器，将 b 发送给辅助服务器

服务器处理阶段：

- 服务器为每个泄露密码的哈希 h 预先生成随机盐值 s_h
- 接收到 a 后，为每个 h 计算 $t=(h \text{ XOR } s_h) \text{ XOR } a$
- 将所有(t,s_h)对组成集合 T 发送给辅助服务器

辅助服务器验证阶段：

- 接收用户发送的 b 和服务器发送的 T
- 对每个(t,s_h)计算 $c= t \text{ XOR } b \text{ XOR } s_h$
- 如果存在 c 为全 0 字节序列，则说明用户密码在泄露集合中

三、程序实现

1、整体结构

实现代码采用面向对象设计，为三个参与方分别创建了对应的类，并提供了核心密码学操作函数：

```
# 核心组件
- 密码哈希函数：hash_password()
- 随机字节生成：generate_random_bytes()
- 字节异或操作：xor_bytes()
- 用户类：User
- 服务器类：Server
- 辅助服务器类：Helper
- 协议模拟函数：simulate_protocol()
```

2、核心代码

密码学基础函数：

- 使用 SHA-256 算法对密码进行哈希处理
- 采用操作系统提供的加密安全随机数生成器
- 实现字节级别的异或操作，这是协议的核心计算基础

User 类：

负责生成用户端的所有数据，包括密码哈希和随机数，以及计算发送给服务器和辅助服务器的中间结果。

Server 类：

管理泄露密码哈希集合，为每个哈希生成随机盐值，并根据用户发送的 a 计算并返回集合 T。

Helper 类：

接收用户的 b 和服务器的 T 集合，执行最终的验证计算，判断用户密码是否泄露。

3、执行流程

程序通过 simulate_protocol()函数模拟整个协议的执行过程，该函数协调三个参与方的交互，最终返回验证结果。

4、测试结果

通过两个测试用例验证了程序的正确性：
测试 1：检查已泄露密码 "qwerty"，返回 True
测试 2：检查未泄露密码 "MySecurePassword123!"，返回 False
测试结果符合预期，证明实现的正确性。

```
==== RESTART: C:/Users/DELL/Desktop/project/Google Password Checkup.py ====
密码 'qwerty' 是否泄露: True
密码 'MySecurePassword123!' 是否泄露: False

==== RESTART: C:/Users/DELL/Desktop/project/Google Password Checkup.py ====
密码 '123456' 是否泄露: True
密码 'MySecurePassword123!' 是否泄露: False

==== RESTART: C:/Users/DELL/Desktop/project/Google Password Checkup.py ====
密码 'letmein' 是否泄露: True
密码 'MySecurePassword123!' 是否泄露: False
```

四、隐私保护机制分析

该协议通过以下机制实现隐私保护：

- **信息分割：**将用户密码哈希的信息分割在 a 和 b 中，分别发送给不同的服务器，单个服务器无法还原出原始哈希
- **随机化处理：**通过引入随机数 r 和盐值 s_h，确保每次交互的中间结果都是不同的，防止通过多次查询进行推断
- **分布式计算：**验证过程需要服务器和辅助服务器的协同计算，任何一方都无法单独完成验证，避免单点隐私泄露
- **零知识证明：**整个过程中，用户不会泄露其密码或密码哈希的任何信息，服务器也不会泄露其存储的泄露密码集合

五、结论

Google Password Checkup 协议通过巧妙的密码学设计和分布式计算模式，成功解决了隐私保护与密码安全检查之间的矛盾。本实现验证了该协议的可行性，为构建更安全的密码检查服务提供了基础。

随着隐私保护技术的不断发展，此类协议将在保护用户数据安全方面发挥越来越重要的作用，平衡安全性与用户隐私之间的关系。