Google Password Checkup协议原理与数学推导

1. 协议概述

Google Password Checkup协议是一种隐私保护的密码检查机制，允许用户检查自己的密码是否在泄露密码列表中，同时不泄露用户密码和具体的泄露密码信息。

该协议涉及三个参与方：

- **客户端(Client)**: 用户侧，拥有待检查的密码
- **服务器(Server)**: 维护泄露密码数据库
- **辅助服务器(Helper Server)**: 协助完成密码检查，与主服务器共同持有秘密份额

2. 核心思想

协议的核心思想是利用秘密共享技术和不经意传输，实现密码的隐私保护检查：

1. 服务器和辅助服务器共同持有泄露密码哈希值的秘密份额
2. 用户将自己密码的哈希值分割为两个份额，分别发送给服务器和辅助服务器
3. 通过密码学协议，在不泄露完整信息的情况下完成比对

3. 数学基础

3.1 符号定义

- p: 一个大素数，作为模运算的基础
- H: 密码哈希函数，将密码映射到有限域 $\mathbb{Z}_p$ 中
- W: 服务器维护的泄露密码集合
- 对于每个泄露密码 $w \in W$，其哈希值为 $h = H(w)$
- 用户密码 w' 的哈希值为 $h' = H(w')$

3.2 秘密共享

秘密共享是协议的核心技术，对于每个泄露密码的哈希值 h:

1. 服务器生成两个份额 s_1 和 s_2，满足:

$$s_1 + s_2 \equiv h \pmod{p}$$

2. 服务器保留 s_1，将 s_2 发送给辅助服务器

对于用户密码的哈希值 h':

1. 客户端生成两个份额 t_1 和 t_2，满足:

$$t_1 + t_2 \equiv h' \pmod{p}$$

2. 客户端保留 t_1，将 t_2 发送给辅助服务器

3.3 协议流程

协议流程可分为以下步骤：

1. **初始化阶段**:
   - 服务器为每个泄露密码 $w \in W$ 计算 $h = H(w)$
   - 服务器将 h 分割为 (s_1, s_2) 并将 s_2 发送给辅助服务器
2. **查询阶段**:
   - 客户端计算 $h' = H(w')$ 并分割为 (t_1, t_2)
   - 服务器生成随机挑战值 $k \in \mathbb{Z}_p^*$
   - 客户端将 t_2 发送给辅助服务器
   - 辅助服务器找到匹配的 s_2（满足 $t_2 = s_2$），生成随机数 r_2，并计算响应:

$$response = (k \cdot s_2 + r_2) \mod p$$

   - 辅助服务器将 r_2 和 $response$ 返回给客户端
   - 客户端计算 $r_1 = (t_1 - r_2) \mod p$
   - 客户端将 $response$ 发送给服务器
   - 服务器验证是否存在 $w \in W$ 使得:

$$(k \cdot s_1 + r_1) \mod p = response$$

   - 如果验证通过，则密码 w' 已泄露

3.4 安全性证明

协议的安全性基于以下事实：

1. **用户隐私保护**:
   - 服务器仅能获得 r_1 和 $response$，无法还原 h' 或 w'
   - 辅助服务器仅能获得 t_2，无法单独确定 h'
2. **服务器数据库保护**:
   - 客户端无法从协议交互中获取任何泄露密码的信息
   - 协议不泄露泄露密码集合的大小或内容
3. **正确性**:
   - 如果 $w' \in W$（密码已泄露），则 $h' = h$ 且 $t_1 + t_2 = s_1 + s_2$
   - 由此可得 $t_1 - s_1 = s_2 - t_2$
   - 辅助服务器计算 r_2 和 $response = k \cdot s_2 + r_2$
   - 客户端计算 $r_1 = t_1 - r_2$
   - 服务器验证 $k \cdot s_1 + r_1 = k \cdot s_1 + t_1 - r_2$
   - 代入 $t_1 = s_1 + s_2 - t_2$ 且 $t_2 = s_2$（因为密码匹配）
   - 可得 $k \cdot s_1 + r_1 = k \cdot s_1 + s_1 + s_2 - t_2 - r_2 = k \cdot s_1 + s_1 - r_2$
   - 进一步化简得 $k \cdot s_1 + r_1 = k \cdot s_2 + r_2 = response$，验证通过