杭州电子科技大学

密码学算法应用实验

一、 实验目的

1.深入理解密码学算法原理和特征。

2.针对现实数据安全防护需求，能够灵活应用基础密码学算法、前沿密码学算法进行安全保护方案的设计。

3.能够综合应用密码学算法的特征，妥善处理密钥分发，存储和使用，并结合不同密码算法的实现，处理不同算法之间输入输出之间的差异带来的挑战，锻炼学生将理论落地为安全防护系统能力。

二、 实验任务

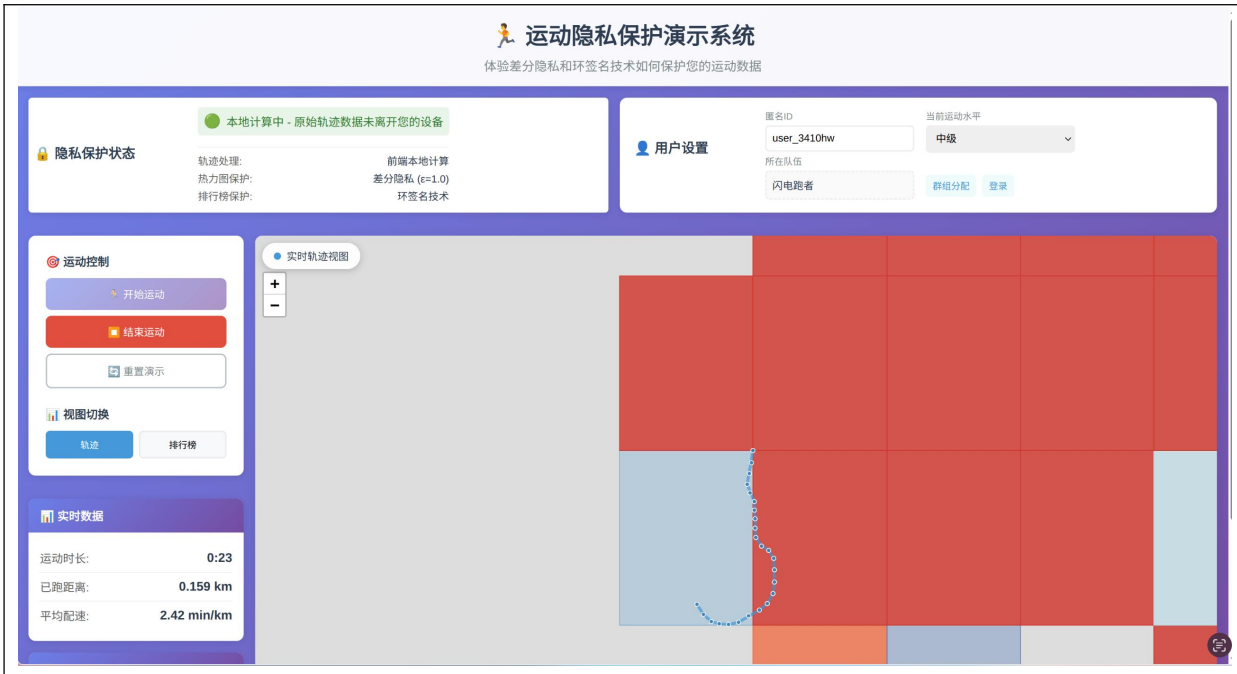
高级实验

按需使用传统和前沿密码学算法，包括对称/非对称密码算法、差分隐私密钥协商算法、环签名技术、同态加密等技术，设计实现一个完整的隐私保护的运动轨迹系统。

三、 密码学算法应用的场景，使用的关键密码算法、安全保护方案的系统架构、关键流程设计

一、系统应用场景概述：

本系统是一个面向运动轨迹与群组竞赛的隐私保护平台。用户使用前端浏览器采集运动轨迹，并实时本地展示，而不将原始轨迹上传到服务器。系统通过差分隐私对运动空间贡献数据进行脱敏汇聚，同时通过环签名实现群组成绩的匿名可信提交。用户无需实名、无需注册账号、无需服务器保存身份信息，即可参与群体活动。这一系统设计使得用户既能贡献运动数据、参与群组竞争，又能确保其个人隐私和身份安全不被暴露。



用户页面总览

二、差分隐私场景与密码技术：

差分隐私用于保护个人运动轨迹数据不被还原。在用户运动过程中，本地前端对连续经纬度进行空间网格化映射，每个网格代表运动区域区块。运动结束时，前端会对每个网格的访问强度叠加拉普拉斯噪声或高斯噪声，从而形成满足 ε -差分隐私约束的扰动数据。扰动后的区块贡献值无法追溯单个用户真实轨迹，但基于大规模用户聚合仍能呈现有效的全局热力分布。在此过程中，原始数据始终在本地留存，服务器得到的永远是匿名化的统计数据。此构架确保即便数据库泄露，也无法重建用户真实路线。

使用的关键技术：

- 差分隐私机制（Laplace 或 Gaussian Mechanism）
- 空间位置扰动与网格化映射

```
export function addLaplaceNoise(actualCount, epsilon = 1.0, sensitivity = 1
    const scale = sensitivity / epsilon;
    const noise = generateLaplaceNoise(scale);
    const noisyCount = actualCount + noise;

    // 确保结果非负
    return Math.max(0, noisyCount);
}

/**
 * 处理轨迹数据，进行区块化和差分隐私加噪
 * @param {Array} trajectory - 原始轨迹点数组
 * @param {number} epsilon - 隐私预算
 * @returns {Array} 加噪后的区块数据 [{x, y, weight}]
 */
export function processTrajectoryWithDP(trajectory, epsilon = 1.0) {
    const gridCounts = {};

    // 1. 区块化：统计每个网格的访问次数
    trajectory.forEach(point => {
        const grid = gpsToGrid(point.lat, point.lng);
        const gridKey = `${grid.x},${grid.y}`;
        gridCounts[gridKey] = (gridCounts[gridKey] || 0) + 1;
    });

    // 2. 差分隐私加噪
    const processedData = [];
    Object.entries(gridCounts).forEach(([gridKey, count]) => {
        const [x, y] = gridKey.split(',').map(Number);
        const noisyWeight = addLaplaceNoise(count, epsilon);

        // 只保留权重显著大于0的区块（优化数据传输）
        if (noisyWeight > 0.1) {
```

差分隐私加噪部分实现代码

三、环签名场景与密码技术：

环签名用于实现匿名可信成绩提交。用户无需上传真实身份，而是通过本地生成公私钥对参与竞赛。系统将具有相似运动水平的用户随机聚合为匿名群组，形成一个环，其特点是每位成员的公钥均参与形成验证集。用户运动结束时，计算个人成绩消息并用自己的私钥和整个环的公钥集合生成环签名。服务器通过环签名验证算法确认成绩来源于环内某一成员，但无法确定是哪一个。这保证了成绩的真实性、不可伪造性与环成员不可区分性。

使用的关键算法：

- ECC-based Ring Signature Scheme
- 随机环公钥集构造

排行榜效果

```python
class RingService:
    def generate_ring(db: Session, user_public_key: str, user_level: str =
        # 1. 从数据库中查找相同水平的其他用户
        other_users = db.query(User).filter(
            User.user_level == user_level,
            User.public_key != user_public_key  # 排除自己
        ).limit(ring_size - 1).all()  # 需要ring_size-1个其他用户

        # 2. 构建公钥列表（包含请求用户自己）
        public_keys = [user_public_key]  # 首先添加请求者
        public_keys.extend([user.public_key for user in other_users])

        # 3. 如果用户数量不足，生成模拟公钥补足环大小
        while len(public_keys) < ring_size:
            mock_keypair = CryptoService.generate_keypair()
            public_keys.append(mock_keypair['public_key'])

        # 4. 随机打乱公钥顺序以增强匿名性
        random.shuffle(public_keys)

        # 5. 生成环ID和趣味群组名
        ring_id = CryptoService.generate_ring_id()
        group_name = random.choice(RingService.GROUP_NAMES)

        # 6. 保存环信息到数据库
        ring = Ring(
            ring_id=ring_id,
            public_keys=public_keys,
            group_name=group_name,
            user_level=user_level
        )
        db.add(ring)
        db.commit()
        db.refresh(ring)
```

环实现部分代码

四、密钥生成与存储方案：

系统不使用传统账户体系，而是采用分布式密钥管理方案。用户在前端浏览器中生成一对非对称密钥，并

在本地安全存储。用户只需将公钥上传给服务器用于环公钥集构建，而私钥永不离开本地存储设备。这样即使服务器遭到入侵，也无法获取任何用户私钥信息或身份映射，是完全无中心信任的架构。

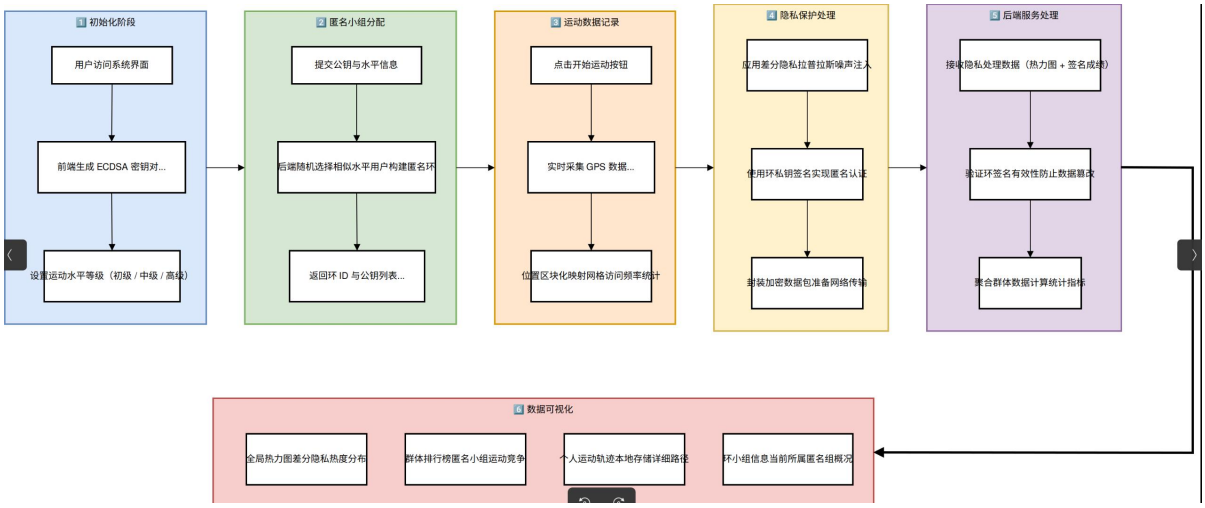
关键要点：

- 密钥对本地生成且私钥只保存在前端

五、安全保护方案的系统架构：

系统采用前端本地敏感计算 + 后端匿名统计结构，保护用户隐私。前端处理精确轨迹、实时绘制和个人成绩计算；后端仅处理匿名差分隐私网格和环签名验证信息。系统架构逻辑上分为三个安全域：本地可信域（用户浏览器）、匿名通信域（前端与后端通信）和聚合统计域（服务器内部热力图和排行榜）。后端从未掌握原始位置、真实身份或运动轨迹，即使数据泄露也无法追踪用户。

六、关键流程设计总结：



系统运行全流程

1. 轨迹采集与本地显示
2. 热力图数据匿名上传
3. 群组匿名匹配与环签名
4. 群组排行榜展示

四、 实验过程中的问题与解决方案总结

在系统的实现过程中，我们遇到了多项技术与设计难题，其中主要包括环签名实现复杂度与密钥管理、地图组件渲染稳定性问题，以及用户体验与隐私保护之间的平衡性挑战。通过针对性优化与架构调整，我们逐渐形成了一套可运行且可验证的隐私保护系统。

1. 环签名实现复杂性与环密钥生命周期管理

在采用环签名替代群签名的方案中，我们发现环签名的落地实现不仅涉及较高的密码学复杂性，还对密钥

生成、分发、更新与撤销提出了新的要求。由于环签名中参与者之间不需相互信任，但仍需在环内正确使用公钥验证路径，这导致密钥生命周期管理变得尤为重要。

为解决此问题，我们采取了以下措施：

- 引入独立的密钥管理模块，负责为每位用户生成唯一的签名私钥与公钥指纹。
- 对环内公钥采用短期有效策略，当超出生命周期后自动更新。
- 在链上存储用户公钥摘要信息，实现可追溯但不可反向识别的密钥状态验证。

通过上述设计，系统既保留了环签名的匿名性优势，也保障了密钥在全生命周期内的可管理性与可撤销性。

2. 地图组件适配

在实验过程中，我们使用地图组件展示匿名上传数据的空间分布信息。实际运行中，地图组件在数据量较大或浏览器渲染性能不足时，容易出现加载失败、渲染卡顿、区块重叠或样式错乱的问题，同时如何使页面地图随用户轨迹移动而动态变化的过程契合用户视觉感知习惯也是我们反复优化的重点。

为解决这一问题，我们从前端渲染优化角度展开改进：

- 将地图渲染逻辑从同步加载优化为延迟加载与按需渲染。
- 引入缓存策略，避免重复生成地图瓦片与标记图层。
- 降低前端实时渲染压力，将部分计算逻辑移至后端预处理。
- 加入页面区块数区间设置，防止过度缩放影响视觉感知。
- 增加强制占位、节流重绘、点迹下采样等机制优化视觉传达。

最终，地图组件稳定性和美观度都得以提升，用户可以流畅地查看经过差分隐私保护的匿名数据分布。

3. 用户实际体验与隐私保护的平衡

系统的初始设计中，我们给予差分隐私与环签名较高优先级，强调数据匿名性与身份不可追踪性。然而，这带来了部分用户体验问题，例如上传延迟、结果模糊化、权限申请复杂等。特别是差分隐私噪声过大时，用户反馈“结果不够精确”。

为应对这一平衡性难题，我们采取了动态隐私保护策略：

- 用户级差分隐私预算采用分层控制，对敏感数据自动提高噪声，对普通数据减弱噪声。
- 对签名与验证操作进行异步处理，不阻塞用户行为链。
- 在不暴露身份的前提下，为用户提供明示性的隐私开关选项。

这些措施有效改善了用户体验，使系统既能满足隐私保护强度要求，又不会损害服务可用性与数据实用性。

五、 实验结果分析以及组员分工

本次实验不仅实现了差分隐私环签名系统的功能性落地，也在开发过程中识别并解决了密码学设计、前端渲染与实际可用性之间的典型冲突问题。系统最终形成了较为完整的隐私保护架构，对匿名可信数据收集系统的实现具有参考意义。

原型设计：沈煜婷 程星芸
后端开发：程星芸
前端开发：倪鑫靓 薛江南