arXiv:2303.02838v1 [cs.DB] 6 Mar 2023

借助安全计算与地理不可区分性，实现准确高效的基于轨迹的接触者追踪

Maocheng Li[1], Yuxiang Zeng[1],Libin Zheng[2],Lei Chen[1,4],和 Qing Li[3]

[1]香港科技大学，中国香港{csmichael,yzengal,leichen}@cse.ust.hk[2]中山大学，中国广州 zhenglb6@mail.sysu.edu.cn[3]香港理工大学，中国香港 csqli@comp.polyu.edu.hk[4]香港科技大学（广州），中国广州

摘要

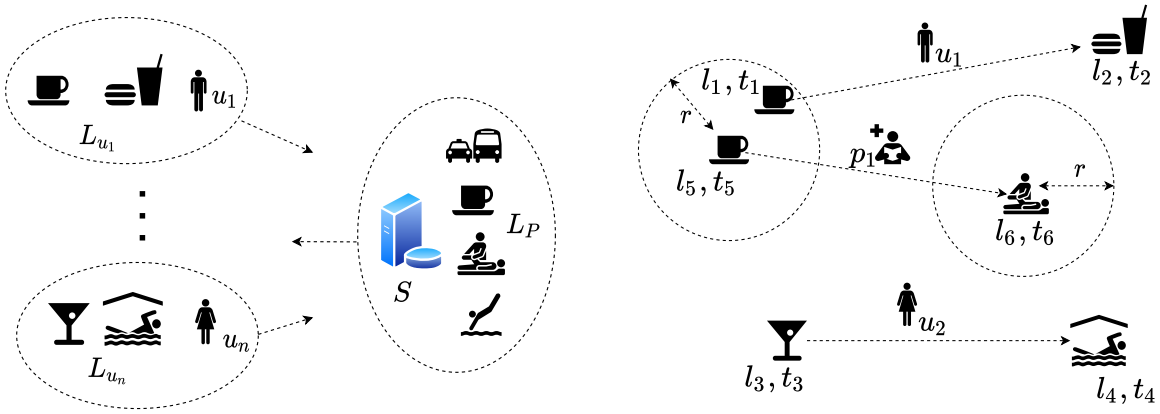
接触者追踪一直被视为限制新冠肺炎等传染病传播的有效措施。基于轨迹的接触者追踪通过比较用户与患者的轨迹，能够追踪直接接触者和间接接触者。尽管轨迹数据被广泛视为敏感的个人数据，但关于如何安全地比较用户和患者的轨迹，以实现准确率高、效率高且具有强隐私保障的接触者追踪，相关研究仍然有限。传统的安全多方计算（MPC）技术运行时间过长，因此无法应用于拥有数百万用户的大城市。在本研究中，我们提出了一种名为 ContactGuard 的技术框架，以实现准确、高效且保护隐私的基于轨迹的接触者追踪。该框架借助 Geo-I（这是针对基于位置的服务（LBS）系统的一种差分隐私概念），仅选取用户的一小部分位置与患者的位置进行比较，从而提高 MPC 基线的效率。大量实验表明，ContactGuard 的运行速度最高可比 MPC 基线快 2.6× 倍，同时不会牺牲接触者追踪的准确率。

关键词： 接触者追踪、差分隐私、空间数据库。

1 引言

接触者追踪一直被视为限制新冠肺炎和埃博拉等传染病传播的关键疫情防控措施之一 [14]。与通常通过访谈和人工追踪开展的传统接触者追踪不同，如今的数字接触者追踪依靠移动设备追踪用户到访的地点，并被认为

[1]版权可能会在未另行通知的情况下转让，转让后此版本将不再可访问。



(a) 我们所研究问题的简化示例。　　(b) 密切接触者的示例。

图 1：(a) 我们所研究问题的简化示例。 L_{u_i} 表示用户 u_i到访的地点。 L_P表示所有患者到访地点的集合。(b) 密切接触者的示例。 u_1 是密切接触者，而 u_2 不是。

更加准确、高效且易于扩展 [19, 21]。以新冠肺炎接触者追踪为例，已有 46 个以上的国家/地区推出了接触者追踪应用 [14], 例如，如新加坡的 TraceTogether 和中国香港特别行政区的安心出行。

作为苹果和谷歌开发的暴露通知等基于蓝牙的接触者追踪应用的补充技术，[2], 基于轨迹的接触者追踪[13, 5]会将用户的轨迹与患者的轨迹进行比较，从而能够追踪直接接触者和间接接触者。当用户和患者在同一时间到访同一地点时，就会产生直接接触。相比之下，间接接触通常发生在用户和患者于不同时间到访同一地点时，此时该地点（例如环境、物体表面）受到污染，成为病毒的传播媒介。由于能够追踪直接接触者和间接接触者，基于轨迹的接触者追踪已在 COVID-19 防控政策较为严格的国家/地区广泛应用。例如，在中国，用户进入餐馆、酒吧和健身房等高风险位置前，需要使用与个人身份关联的健康码进行登记。

尽管基于轨迹的接触者追踪很有用，但轨迹数据的隐私显然是一个问题。事实上，隐私顾虑阻碍了接触者追踪应用在香港等城市得到广泛采用；甚至出现了极端情况：市民使用两部不同的手机，一部日常使用，另一部则专门用于满足进入高风险位置时的签到要求，以防止隐私泄露。现有的隐私保护的基于轨迹的接触者追踪研究依赖特定硬件（例如英特尔 SGX [13] 等可信硬件），但其安全保障高度依赖硬件，导致可移植性较差。事实上，英特尔计划从其第 11 代和第 12 代处理器开始停止支持 SGX。因此，在本研究中，我们旨在开发一种独立于硬件的软件解决方案，以实现高接触者追踪准确率、出色的执行效率和强隐私保障。

本文提出了隐私保护接触者追踪（*PPCT*）问题，并通过图 1a 中的简化示例加以说明。我们的目标是准确判断每位用户 u_i 是否为密切接触者。如果用户 u_i 在某些时空约束下与某位患者到访过同一地点（例如，相距 5 米以内到访同一地点，且

[2]https://covid19.apple.com/contacttracing

处于 2 天的时间窗口内），那么我们提出的解决方案应能准确识别该用户u_i为密切接触者。PPCT 面临的关键挑战在于其对隐私有严格要求：在接触者追踪的整个过程中，其他参与方都无法访问用户的私人信息（包括用户到访的地点及相应时间戳），从而确保这些信息得到严格保护。

PPCT 面临的挑战不止于强隐私保护要求。一方面，接触者追踪应用需要具备较高的准确率。以新冠肺炎为例，一些国家投入大量资源来识别患者可能的密切接触者，有时仅在某个地区发现一例患者，该地区全体市民就需要接受强制检测。因此，我们提出的解决方案需要尽可能避免假阴性，并保持相当高的召回率，最好接近 100%。另一方面，我们旨在为 PPCT问题提供高效的解决方案，因为传统的安全多方计算（MPC）技术通常会导致运行时间长到难以实际应用。正如实验结果所示，对于 100 万用户（相当于一座现代城市的人口），一种朴素的 MPC 解决方案需要 5 天以上。如此长的运行时间使其无法每天执行，因为甚至不能在 24小时内完成。

为应对上述挑战，我们提出了一种名为 Con- tactGuard 的新型技术框架。ContactGuard 通过仅选取用户位置中的少量子集与患者位置进行比较，提高 MPC 操作的效率。子集选择过程借助一种高效的隐私保护机制——基于位置的服务（LBS）系统中的地理不可区分性（Geo-I）来完成。其基本思路是，每位用户利用 Geo-I 对自己的真实访问位置进行扰动，并仅将扰动位置提交给存储患者位置的服务器。服务器使用用户的扰动位置与患者位置进行比较，然后告知用户哪些位置的“风险”更高，因为这些位置距离患者更近。之后，用户只需对高风险位置使用 MPC，从而大幅降低计算开销。

总而言之，本文的贡献如下：

- 我们在第2节中正式定义了一个重要问题——隐私保护接触者追踪（PPCT）问题，该问题旨在解决基于轨迹的接触者追踪中的隐私问题。

- 我们在第3节中提出了一种新颖的解决方案 ContactGuard。该方案结合了两种不同隐私保护范式的优势：差分隐私和安全多方计算。由于在 MPC操作期间仅选择少量位置与患者的位置进行比较，运行时间得到了显著改善。

- 我们在第4节中开展了大量实验，以验证 ContactGuard 的有效性和效率。其运行速度最高比 MPC 基线快 2.6× 倍，同时不会牺牲接触者追踪的准确率。在为每位用户设置适中的隐私预算时，ContactGuard 的召回率和精确率均接近 100%。

此外，我们在第5节中回顾了相关工作，并在第6节中进行了总结。

2 问题定义

本节介绍一些基本概念、对手模型，以及隐私保护接触者追踪（PPCT）问题的形式化定义。

2.1 基本概念

定义 1（位置）。 位置 $l =$ （x, y）表示欧几里得空间中坐标为 （x, y）的二维空间点。

定义 2 (轨迹**)**。 轨迹是一组（位置，时间戳）元组，用于指示到访地点及其到访时间。轨迹 $L = \{(l_1, t_1) \ldots, (l_{|L|}, t_{|L|})\}$，其中 $|L|$ 是轨迹中的访问位置数量。

示例：设位置 l_1 =(300,500) 和 l_2 为另一个位置 l_2 =(200, 200)。轨迹 $L_{example} = \{(l_1,$ 2020-06-10 12:28:46),(l_2, 2020-06-10 17:03:24)} 表示位置 l_1 在时间戳 2021-06-10 12:28:46 被到访，位置 l_2 在时间戳 2021-06-10 17:03:24 被到访。 $L_{example}$ 的大小为 2，即， $|L_{example}| = 2$。

这里，轨迹 的定义既包括较短的序列（例如，用户早上离开家，前往工作地点，晚上再回家），也包括跨越多天的较长序列。在我们的问题设定中，为了追踪接触者，用户的轨迹包含该用户在疾病潜伏期内到访的所有位置的并集（例如，COVID-19 [12, 10] 的过去 14 天）。

定义 3（用户）。 集合 U 表示所有 n 用户。每个用户 $u \in U$ 都关联一条轨迹 $L_u = \{$（l_1, t_1） $\ldots$，（$l_{|L_u|}, t_{|L_u|}$）}；根据定义 *2*，该轨迹表示用户 u 到访的位置及到访时间。

需要将每个用户与患者进行比对，以确定该用户是否是某位患者的接触者。患者的定义与用户类似；为了便于说明接触者追踪问题，我们将二者分别定义。

定义 4（患者）。 集合 P 表示所有 m 患者。每位患者 $p \in P$ 都有一条轨迹$L_p = \{$（l_1, t_1）.....（$l_{|L_p|}, t_{|L_p|}$）}，遵循定义 *2*，表示患者 p到访的位置及到访时间。集合 $L_P = L_{p_1} \cup L_{p_2} \cup \ldots \cup L_{p_m}$ 是所有患者 p_1，.....， p_m的轨迹的并集。

在我们的问题设定中，我们使用 L_P 表示所有患者轨迹的聚合集合。这是一个重要概念，因为如果用户 $u \in U$ 的轨迹 L_u 与 L_P中的某个位置重叠，则该用户 u被定义为接触者 （更正式的定义见定义5）。无需识别具体是哪位患者 $p \in P$ 导致了该接触。实际上，正如我们将在第 2.2 节的对手模型和系统设置中进一步解释的那样，只需存储所有患者的轨迹 L_P 的并集，而无需区分每位患者的个人轨迹，这也能增强隐私保护。

定义5（接触者）。 给定距离阈值 r 和时间差阈值 δ，如果用户 $u \in U$ 到访的位置 l_u 与某位患者 $p \in P$ 到访的某个位置 l_p 的距离不超过 r，且两次到访的时间差不超过 δ，则用户 u 被称为接触者，即：

$$\exists (l_u, t_u) \in L_u \; \exists (l_p, t_p) \in L_P \; ((d_s(l_u, l_p) \leq r) \wedge (d_t(t_p, t_u) \leq \delta)) \tag{1}$$

其中，函数 d_s（l_u, l_p） 表示空间距离，即位置 l_u 与位置 l_p 之间的欧几里得距离。d_t（t_u, t_p） 表示时间差，即从较早的时间戳 t_p 到较晚的时间戳 t_u 之间的秒数。

我们在示例 1 中给出一个简单示例，以更好地说明接触者的概念。

示例 *1.* 如图*1b*所示，有一名患者 p_1 和两名用户 u_1-u_2。用户 u_1 的轨迹为 $L_{u_1} = \{(l_1, t_1)、(l_2, t_2)\}$。用户 u_2 的轨迹为 $L_{u_2} = \{(l_3, t_3)、(l_4, t_4)\}$。患者集合 P（仅包含一名患者 p）的轨迹为 $L_P = \{(l_5, t_5)、(l_6, t_6)\}$。图中展示了各个位置，下面我们进一步明确时间。设 $t_1 = 2021\text{-}06\text{-}10\ 11{:}00{:}00$，$t_5 = 2021\text{-}06\text{-}10\ 10{:}00{:}00$，这表示患者 p 到达位置 l_5 的时间比用户 u 到达 l_1早一小时。

根据定义*5*，设图中所示的距离为 r ，时间差阈值为 $\delta = 2$ 小时，则用户 u_1 是接触者，因为 d_s（l_1，l_5）$\leq r$，即，u_1 到访过l_1，且 l_1 与患者 p_1的到访地点 l_5之间的距离在 r 以内，并且 d_t（t_1，t_5）$= 1$ 小时 $\leq \delta$，即，两次访问之间的时间差在 δ （$\delta = 2$ 小时）以内。

另一方面，用户 u_2 不是密切接触者，因为她到访过的位置 l_3 和 l_4均不在任何患者到访地点的邻近范围内。

2.2 对手模型

我们的应用中有两个主要角色：用户（客户端）和政府（服务器）。患者的所有轨迹（如定义4中的 L_P 所表示）都会在服务器上汇总并存储。该设置遵循现实世界的情况：政府通常会收集确诊患者的行踪，以尽量减少进一步传播，通常首先追踪患者的密切接触者。例如，香港的相关法律已对此作出规定[3]。

在我们的问题中，我们采用半诚实模型作为对手模型。对手可能存在于客户端（用户）和服务器端（政府）。我们假设客户端和服务器都对其他参与方的私人信息感兴趣，但并无恶意，并会遵循我们设计的系统协议。半诚实模型是近期隐私保护位置服务相关应用中普遍采用的一种设定 [24, 23]。

2.3 隐私保护接触者追踪问题

基于前文介绍的概念和对手模型，我们现在将隐私保护接触者追踪（PPCT）问题定义如下。

定义6（隐私保护接触者追踪（PPCT）问题）。 给定一组 U 用户、每位用户 $u \in U$的轨迹 L_u 、一组 P 患者、一组 L_P （其中包含所有患者 p_1、.....、 $p_m \in P$的轨迹并集）、距离阈值 r以及时间差阈值 δ，*PPCT*问题需要正确判定每位用户是否为接触者（见定义*5*）。

此外，*PPCT* 还有以下隐私要求：

- 计算过程需要针对每个用户的轨迹 L_u 实现差分隐私 / 保密性关于。
- 计算过程需要在相对于 L_P（即患者轨迹的并集）方面实现差分隐私/保密性。

[3] https://www.elegislation.gov.hk/hk/cap599D!en?索引 CS=N

3 方法论

本节首先介绍两种基线方法：一种基于安全多方计算（MPC），另一种基于地理不可区分性（Geo-I）。然后介绍我们提出的方法——ContactGuard。

3.1 基线方法

MPC 和 Geo-I 是满足 PPCT问题中隐私要求的两种不同范式，但它们在效率和准确率方面差异很大。MPC 允许服务器和客户端使用密码学原语，在不向其他参与方泄露任何一方精确位置的情况下，安全地比较每个到访地点。然而，这会带来显著的计算开销。

Geo-I [1] 是将差分隐私扩展到空间领域的一种概念（严格来说，Geo-I 是本地差分隐私概念的一种变体 [8, 3]）。它是一种可高效应用于位置输入的机制，但 Geo-I 会向输入中注入噪声（通过将输入位置扰动为混淆位置来保护输入位置）。随后，使用扰动位置比较轨迹会导致错误，并降低接触者追踪的准确率。

3.1.1 MPC 基线

MPC 基线是将现有安全多方计算技术直接应用于我们的问题。在我们的设置中，每次都将一个客户端（用户）和服务器（持有患者数据）视为两个参与方。每个参与方都持有各自的访问位置，这些位置是私有的。然后，他们使用 MPC 操作比较各自的访问位置，并根据定义5检查该用户是否为接触者。请注意，该基线计算的是精确结果，这意味着它不会损失任何准确率。

3.1.2 Geo-I 基线

在 Geo-I 基线中，每个用户使用 Geo-I（算法 1）将自己的轨迹 L_u 扰动为受保护的含噪位置集 L'_u。然后，用户将 L'_u 提交给服务器。服务器直接将L'_u 与患者位置 L_P进行比较。如果存在某个扰动位置 $l' \in L'_u$，其距离在 r'（一个系统参数）以内，那么 u 就会被识别为接触者。

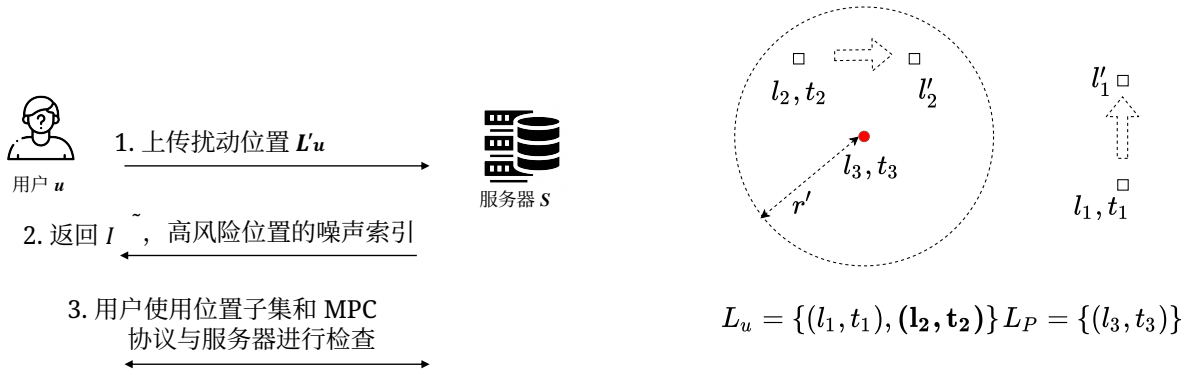
算法 1： 扰动位置集 _ _

输入： ϵ，L_u。

输出： L'_u。

1 $L'_u := \{\}$
2 $\epsilon' = \epsilon / |L_u|$
3 **对** $l \in L_u$ **执行循环**
4 $l' =$ Geo-I(ϵ'，l)
5 L'_u.插入(l')
6 **返回** L'_u

算法 1 是一种通用方法：给定一组原始位置 L_u 和总隐私预算 ϵ，生成一组扰动位置 L'_u。在第 2 行，总隐私预算 ϵ 被平均分成 $|L_u|$ 份。因此，每份为 $\epsilon' = \epsilon/|L_u|$。然后，我们通过 Geo-I 机制将每个位置 $l \in L_u$ 扰动为一个新位置 l'。所有扰动位置 l' 组成扰动位置集合 L'_u。



(a) 总体工作流程。　　(b) 子集选择。

图 2：步骤 3 中的总体工作流程和子集选择过程。

请注意，在定义 2 中对轨迹 L_u 的原始定义里，每个到访地点 l 都关联着该地点被到访时的时间戳（见示例 1）。我们省略每个到访地点的时间戳信息，仅 将位置扰动为扰动位置。我们滥用符号 L_u 和 L'_u，分别表示仅包含空间位置（定义 1）的轨迹以及生成的扰动位置。

时间复杂度。 对于算法 1（客户端），时间复杂度为 $\mathcal{O}(|L_u|)$，相对于输入 L_u 中给出的访问位置数量呈线性增长。对于服务器端，由于它会将 L'_u 中的每个位置与 L_P 中的每个位置进行比较，时间复杂度为 $\mathcal{O}(|L'_u||L_P|) \rightarrow \mathcal{O}(|L_u||L_P|)$。因此，服务器 S 完成所有用户的处理所需的总时间复杂度为 $\mathcal{O}(\sum_u |L_u||L_P|) \rightarrow \mathcal{O}(|U||L_P| \max_u |L_u|)$。

隐私分析。 我们将算法 1 的隐私分析留待第 3.2 节讨论，因为它与 ContactGuard 的隐私分析相同。

3.2 我们的解决方案 ContactGuard

本节提出一种名为 ContactGuard 的新方法。如前所述，MPC 基线方法具有出色的准确率，因为它会执行计算开销很大的 MPC 操作，将用户的所有访问位置与患者的访问位置进行比较。MPC 基线方法的弱点是效率低下。另一方面，Geo-I 基线方法效率较高，因为每个客户端（用户）都会应用高效的 Geo-I 机制，将其访问位置转换为扰动位置。但扰动位置不够准确，可能会导致 PPCT 问题出现假阳性/假阴性。

ContactGuard 旨在结合两种基线方法的优点。一方面，它使用精确位置和 MPC 来判断用户是否为接触者，从而在提供强隐私保护的同时确保出色的准确率。另一方面，它通过仅选择用户访问位置的一个子集来执行开销很大的 MPC 操作，从而加快 MPC 操作。该选择过程借助 Geo-I 完成。图 2a 展示了 ContactGuard 的总体工作流程。接下来将介绍具体细节。

步骤 1. 位置扰动。 在此步骤中，用户将其访问位置集合 L_u 扰动为扰动后的访问位置集合 L'_u，并将 L'_u 提交给服务器。这些步骤与 Geo-I 基线（第 3.1.2 节）类似。

给定每个用户的隐私预算 ϵ 和真实位置集合 L_u，用户使用算法 1 生成扰动后的位置集合 L'_u。与 Geo-I 基线类似，总隐私预算ϵ 平均分为 $|L_u|$ 份。随后，每个位置 $l \in L_u$ 都通过 Geo-I 机制扰动为一个噪声位置

l' 使用隐私预算为 $\epsilon' = \epsilon/|L_u|$的 Geo-I 机制。

与 Geo-I 基线类似，我们滥用符号 L_u 和 L'_u，分别仅表示位置（不含时间戳）和扰动位置。这与定义 2 中访问位置集合 L_u 的原始定义略有不同，后者的每个位置都与一个时间戳相关联。在扰动过程中，我们完全忽略所有时间信息，以此为时间戳提供强隐私保障（因为我们根本不使用时间戳）。在后续的 MPC 步骤中（步骤 3），我们将使用时间信息检查时间约束，以确定接触者。

隐私分析。为了了解算法 1 的隐私保障程度，我们对其进行理论隐私分析。它将 Geo-I [1] 从保护单个位置扩展到保护一组位置。我们在 [1], 中提到了该扩展的基本思想，本文将在此给出具体且形式化的分析。

首先，我们将 Geo-I 的隐私定义从单个位置扩展到一组位置。

定义 7。*(General-Geo-I)* 对于两个位置元组 $\mathbf{l} = (l_1, \ldots, l_n)$, $\mathbf{l}' = (l'_1, \ldots, l'_n)$、隐私参数 ϵ，机制 M 满足 ϵ-*General-Geo-I*，当且仅当：

$$d_\rho(M(\mathbf{l}), M(\mathbf{l}')) \leq \epsilon d_\infty(\mathbf{l}, \mathbf{l}'),$$

其中 $d_\infty(\mathbf{l},\mathbf{l}') = max_i d(l_i, l'_i)$，即$\mathbf{l}$ 与$\mathbf{l}'$中所有位置对之间的最大欧几里得距离。

与针对单个位置的 Geo-I 定义不同，此处两个位置元组之间的距离定义为任意维度上的最大距离。如果随机机制 M 满足定义 7 中的 General-Geo-I，那么对于其输出的任意扰动后的位置集合 $\mathbf{o} = (o_1, \ldots, o_n)$（这是由 M生成的一系列扰动位置），任何对手都无法区分这些扰动位置是由真实位置集合$\mathbf{l}$ 还是$\mathbf{l}'$生成的。对于真实位置集合$\mathbf{l}$ 或其相邻输入$\mathbf{l}'$，生成相同输出$\mathbf{o}$ 的概率之比上界为 $\epsilon d_\infty(\mathbf{l},\mathbf{l}')$。

作为一个特例，当$\mathbf{l}$中只有一个位置时，定义 7 就变成了 Geo-I 中的原始定义。我们首先通过证明 General-Geo-I 的组合定理来开展隐私分析。

引理 1. 设 K_0 是一个满足 ϵ_1-*Geo-I* 的机制，K_1 是另一个满足ϵ_2-*Geo-I* 的机制。对于给定的双位置元组 $\mathbf{l} = (l_1, l_2)$，将 K_0 和 K_1 组合为$K_{1,2}(\mathbf{l}) = (K_1(l_1)\ K_2(l_2))$ 后，满足$(\epsilon_1 + \epsilon_2)$-*General-Geo-I*（定义 7）。

证明。设组合机制的一个潜在输出 $K_{1,2}(\mathbf{l})$ 为 $\mathbf{o} = (o_1, o_2)$，表示一个由两个扰动位置组成的元组。由两个输入位置集合$\mathbf{l} = (l_1, l_2)$ 和 $\mathbf{l}' = (l'_1, l'_2)$ 生成 $\mathbf{o}$ 的概率分别为 $\Pr[K_{1,2}(\mathbf{l}) = \mathbf{o}]$ 和 $\Pr[K_{1,2}(\mathbf{l}') = \mathbf{o}]$,。然后，我们计算这两个概率的比值：

$$\frac{\Pr[K_{1,2}(\mathbf{l}) = \mathbf{o}]}{\Pr[K_{1,2}(\mathbf{l}') = \mathbf{o}]} = \frac{\Pr[K_1(l_1) = o_1] \cdot \Pr[K_2(l_2) = o_2]}{\Pr[K_1(l'_1) = o_1] \cdot \Pr[K_2(l'_2) = o_2]} \tag{2}$$

$$= \frac{\Pr[K_1(l_1) = o_1]}{\Pr[K_1(l'_1) = o_1]} \cdot \frac{\Pr[K_2(l_2) = o_2]}{\cdot \Pr[K_2(l'_2) = o_2]} \tag{3}$$

$$\leq e^{\epsilon_1 \cdot d(l_1, l'_1)} \cdot e^{\epsilon_2 \cdot d(l_2, l'_2)} \tag{4}$$

$$\leq e^{\epsilon_1 \cdot d_\infty(\mathbf{l},\mathbf{l}')} \cdot e^{\epsilon_2 \cdot d_\infty(\mathbf{l},\mathbf{l}')} \tag{5}$$

$$= e^{(\epsilon_1 + \epsilon_2) d_\infty(\mathbf{l},\mathbf{l}')} \tag{6}$$

公式（6）的最终结果表明，$K_{1,2}$ 满足 $(\epsilon_1 + \epsilon_2)$-General-Geo-I（定义 7）。

□

定理 1. 算法 *1* 满足 ϵ-*General-Geo-I*（定义 7）。

证明。 我们使用引理 1 中 Geo-I 的组合定理，证明算法 1 具有线性组合性质，并且在整个位置集合上消耗的隐私预算总量为 ϵ。

在算法 1 的第 4 行，对于每个位置 $l \in L_u$，都会将其扰动为 l'，隐私预算为$\epsilon' = \epsilon_u/|L_u|$。然后，对于每个位置，我们都能实现 ϵ'-Geo-I。若 L_u中有两个位置，由于每个位置都是独立扰动的，我们便能实现 $\epsilon' + \epsilon' = 2\epsilon'$-General-Geo-I。接着，通过归纳法，对于 $|L_u|$ 个位置，我们能实现 $\sum_{l \in L_u} \epsilon' = \epsilon' \cdot |L_u| = \frac{\epsilon}{|L_u|} \cdot |L_u| = \epsilon$-General-Geo-I。

□

步骤 2。子集选择。在步骤 1 之后，每个用户都有一组扰动位置 L'_u，并将其提交给服务器。在步骤 2（子集选择）中，服务器收到 L'_u 后，根据扰动位置选择一个高风险位置子集。此步骤在服务器端执行，患者的真实位置存储在服务器端。服务器将所选高风险位置的索引返回给客户端（例如，如果第 2 个位置和第 4 个位置属于高风险位置，则服务器返回 $\{2, 4\}$），以指示 L_u 中哪些位置接近患者。为了保护患者的隐私，使用随机响应对高风险位置的索引进行扰动，生成噪声索引。

在服务器端，服务器不知道访问的真实位置和时间戳，这些信息存储在客户端（用户）的 L_u 中。然而，服务器可以访问患者访问过的真实位置，这些位置表示为 L_P。 L_P 是判断用户位置是否属于高风险位置的依据。

定理 2. 子集选择步骤满足 ϵ_P-本地差分隐私。

我们还在其中为这一步提供了一个详细的运行示例。

~步骤 3. 加速 MPC。 步骤 2 之后，用户 u 会收到集合 I，其中包含服务器识别出的高风险位置的索引。在步骤 3（加速 MPC）中，用户使用多方计算协议与服务器核实自己是否为接触者，~并且仅使用集合 I所指示的位置。相比之下，MPC 基线会使用L_u 中的每个位置，通过多方计算协议与服务器核实。这里，用户仅使用一小部分位置来执行安全计算。

我们的方法运行与 MPC 基线相同的安全计算流程。不过，它仅将位置子集作为输入提供给 ProtocolIO 对象 io。时间戳也包含在 io 对象中，并用于与患者轨迹L_P进行比较。这与 Geo-I 基线或 ContactGuard 中步骤 1 的位置扰动不同，后者仅使用空间信息（位置）。

总体分析。 总体而言，服务器完成所有用户的处理需要 $\mathcal{O}(\sum_u |I_u||L_P|) \rightarrow \mathcal{O}(|U||L_P| \max_u |I_u|)$，其中 I_u表示每个用户 u 对应的子集。

隐私分析。 ContactGuard 为 L_u 和 L_P提供端到端隐私保证关于。。

表1列出了两种基线方法与我们提出的 ContactGuard 的对比。在实现相同隐私和准确率的同时，我们提出的方案显著提高了 MPC 方案的效率。

表1：基线方法与 ContactGuard 的比较

方法	MPC 基线	Geo-I 基线	ContactGuard
准确率（召回率）	100% √	39.0-70% ×	86-100% √
效率	安全操作次数：$\mathcal{O}$（$\|U\|\|L_P\|\max_u \|L_u\|$）×	明文操作数：$\mathcal{O}(\|U\|\|L_P\|\text{最大}_u \|L_u\|)$ √	安全操作数：$\mathcal{O}(\|U\|\|L_P\|\text{最大}_u \|I_u\|)$ √
隐私	机密性 √	本地-d 差分隐私 √	本地-d 差分隐私 √

4 实验研究

本节首先介绍第 4.1 节中的实验设置，然后在第 4.2 节中展示详细的实验结果。

4.1 实验设置

数据集。 我们在实验中使用了真实世界数据集 Gowalla [2] 和随机生成的合成数据集。

基线方法。 我们将提出的 ContactGuard（简称**CG**）方法与 MPC 基线方法（第 3.1.1 节，简称**MPC**）和 Geo-I 基线（第 3.1.2 节，简称**Geo-I**）进行比较。
指标。 我们关注以下指标：

- 召回率：找到的真实密切接触者数量除以密切接触者总数。
- 精确率：找到的真实密切接触者数量除以预测的近距离接触者数量。
- F_1 得分：精确率和召回率的调和平均值。 $F_1 = 2 \cdot \frac{\text{精确率} \cdot \text{召回率}}{\text{精确率} + \text{召回率}}$。
- 准确率：正确分类的用户所占的百分比。
- 运行时间：方法的运行时间（秒）。

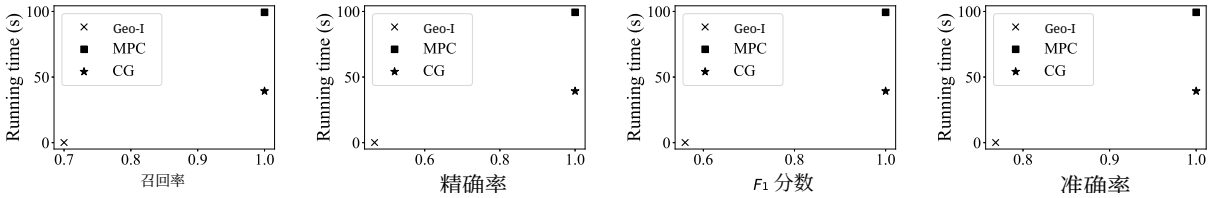
召回率被视为我们实验中最重要的指标，因为它衡量了所测试方法从真实的密切接触者中识别潜在密切接触者的能力，这对于传染病接触者追踪的应用场景至关重要。

控制变量。 我们在实验中改变以下控制变量，以测试我们的方法。默认参数以粗体显示。隐私预算采用实际部署（苹果和谷歌）中的常见设置 [6]。

- 用户数量： $|U| \in [\mathbf{200}, 400, 800, 1600]$。可扩展性测试： $|U| \in [10K, 100K, 1M]$。
- 每位用户的隐私预算： $\epsilon \in [2.0, 3.0, \mathbf{4.0}, 5.0]$。
- 患者的隐私预算： $\epsilon_P \in [2.0, 3.0, \mathbf{4.0}, 5.0]$。

4.2 实验结果

结果表明，所提出的解决方案 ContactGuard 在有效性与效率之间实现了最佳权衡，将 MPC 基线的运行时间最多缩短了 $2.85\times$。然而，高效的 Geo-I 基线仅能提供较差的有效性。与此同时，ContactGuard 在有效性方面（通过召回率、精确率、 F_1 分数和准确率衡量）与



(a) 召回率 (b) 精确率 (c) F_1 得分 (d) 准确率

图 3：Gowalla 数据集上不同方法的有效性（召回率/精确率/F_1/准确率）与运行时间之间的权衡。 $|U| = 200,\ \epsilon = 4.0.$

MPC 基线保持相同水平。可扩展性测试还表明，ContactGuard 显著加快了 MPC 基线的运行速度。当用户数量较大时（例如，500K），ContactGuard 支持每日执行，这对于实际应用十分理想；而 MPC 基线则无法在合理时间内完成（例如，24小时）。

有效性与效率权衡。图 3 展示了不同方法在有效性与效率之间的权衡。y 轴表示不同方法（MPC 基线、Geo-I 基线和我们提出的 ContactGuard）的运行时间，x 轴表示相应的有效性指标（召回率、精确率、 F_1 和准确率）。

如图 3 所示，MPC 基线是一种精确方法，在所有有效性指标上均达到 100%。然而，高准确率伴随着难以承受的运行时间。当用户数量为 200（$|U| = 200$）时，运行时间为 99.35 秒；当用户数量为 400（$|U| = 400$）时，运行时间为 194.99 秒。

相比之下，Geo-I 方法效率较高，但有效性显著下降。对于设置 $|U| = 200,$ $\epsilon = 4.0$，Geo-I 的召回率约为 70%，精确率仅为 46.7%。对于设置 $|U| = 400,$ $\epsilon = 3.0$，召回率降至 39.0%，精确率降至 42.0%。

我们提出的 ContactGuard 实现了最佳的有效性/效率权衡。对于设置 $|U| = 200,\ \epsilon = 4.0$（图 3），CG 的有效性与 MPC 基线相当，在召回率、精确率和准确率方面均达到 100%。

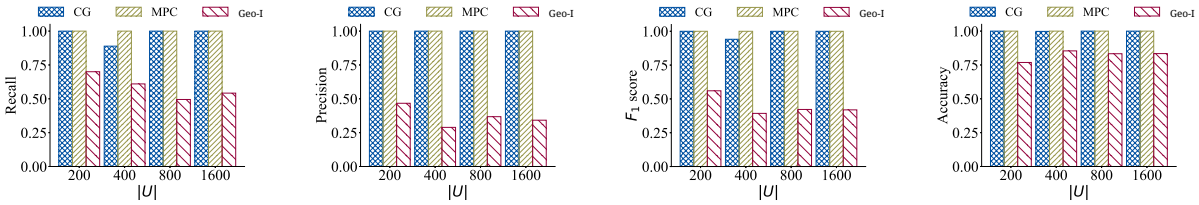
与 Geo-I 基线相比，CG 方法在召回率、精确率和准确率方面分别提升了 2.28×、2.38× 和 1.24×。另一方面，与精确的 MPC 基线相比，CG 方法的速度约快 2.5× 倍。

控制变量的影响。 随后，我们测试了不同输入规模 ($|U|$) 和不同隐私预算值 (ϵ) 下的有效性。

改变 $|U|$: 图 4 展示了在 Gowalla 数据集中改变用户数量 ($|U|$) 时的有效性指标（召回率、精确率、 F_1 和准确率）。在不同输入规模下，CG 方法的召回率/精确率/F_1 得分/准确率均与精确的 MPC 基线在不同 $|U|$ 下保持相同水平。这验证了我们的 CG 方法在不同输入规模下具有很高的有效性。

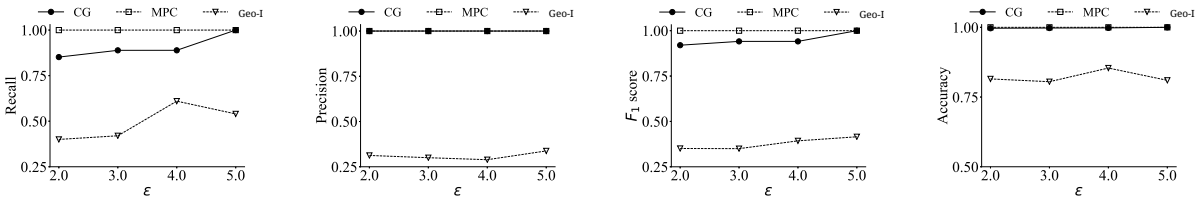
另一方面，Geo-I 基线在不同 $|U|$ 下牺牲了显著的有效性（召回率/精确率/F_1/准确率）。当 $|U| = 200$时，召回率从 70% 降至 $|U| = 1600$时的 54.2%。精确率始终低于 50%，且当 $|U|$大于 800 时徘徊在 35% 左右。这表明，纯粹基于差分隐私的解决方案会向扰动位置注入噪声，而扰动位置不可避免地会导致接触者追踪应用的有效性较差。

ϵ变化：图 5 展示了隐私预算对有效性（召回率/精确率/F_1/准确率）的影响。在召回率方面，当 ϵ 时，CG 达到了与 MPC 基线相同的有效性水平



(a) 召回率。 (b) 精确率。 (c) F_1 得分。 (d) 准确率。

图 4：在 Gowalla 数据集中改变用户数量（$|U|$）时的有效性（召回率/精确率/F_1/准确率）。 $\epsilon = 4.0$.



（a）召回率。 （b）精确率。 （c） F_1 得分。 （d）准确率。

图 5：在 Gowalla 数据集中改变隐私预算（ϵ）时的有效性（召回率/精确率/F_1/准确率）。 $|U| = 400$.

（达到 100%）。当 $\epsilon = 5.0$ 时，召回率为 85.2%；隐私预算较小（$\epsilon = 2.0$）时，召回率为 85.2%，而当 $\epsilon = 3.0$ 时则提高至 88.9%。在这两种设置下，相比 Geo-I 基线，提升约为 2.12×。

就精确率而言，CG 不会产生假阳性，因为只有当用户确实存在与患者重叠的真实访问位置（该位置包含在所选子集中，并经过 MPC 操作验证）时，才会返回阳性结果。因此，在不同 ϵ下，精确率始终保持在 100%。

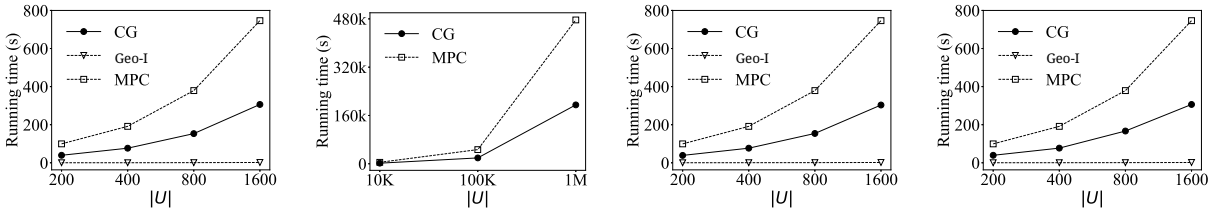
F_1 分数呈现出与召回率相似的变化趋势，因为它是召回率和精确率的平均值。CG 的准确率在不同 ϵ下均保持在 99% 以上。

效率和可扩展性。 与 MPC 基线相比，我们的 CG 方案显著提高了加速比。图6展示了在用户数量扩展至 100 万时，不同方法在不同输入规模下的运行时间。

当用户数量（$|U|$）从 200 增至 1600 时，MPC 基线的运行时间从 99.35 秒增至 746.34 秒。运行时间随$|U|$线性增长。

相比之下，共轭梯度法的运行时间从39.31秒增至306.20秒，也随 $|U|$线性增长。当 $|U|$ 从200增加到1600时，共轭梯度法相对于MPC的加速比为2.52×、2.50×、2.48×和2.44× 。结果如图6a所示。

随后，我们使用合成数据集比较 MPC 基线和 CG 方法的可扩展性。结果如图6b所示。对于 $|U|$ =10K、100K 和 1M，MPC 的运行时间分别为 4754.7 秒、46,913 秒（13 小时）和 476,504.6 秒（5.5 天）。相比之下，CG 方法的运行时间分别为 1936.8 秒、19,087.2 秒（5.3 小时）和 195,060.7 秒（2.25 天）。相较于 MPC 基线，它保持了 $2.5\times$ 加速比。这验证了 CG 中的子集选择显著降低了多方计算协议的运行时间。当用户数量达到 100 万时，MPC 基线需要约 5.5 天来处理所有用户，而我们提出的 CG 可在约 2 天内完成。



(a) $\epsilon = 4.0$。 (b) 可扩展性测试。 (c) $\epsilon = 2.0$。 (d) $\epsilon = 3.0$。

图6：在合成数据集上改变用户数量（$|U|$）时的效率（运行时间，单位：秒）。

5 相关工作

我们的工作与 REACT [5], 密切相关，因为它同样使用 Geo-I [1] 来保护位置隐私。在 REACT 中，每个位置均采用具有相同隐私预算的 Geo-I。如实验所示（无论是在 [5]中还是在我们的结果中），Geo-I 基线在召回率和精确率方面的表现都较差，因为它仅使用扰动位置。作为与 Geo-I 类似的概念，本地差分隐私也被用于保护轨迹数据 [4]。然而，它的主要用途是在相对较大的区域上进行聚合统计数据，而在我们的应用中，每位用户的接触者追踪和细粒度到访地点都很重要。

接触者追踪还使用了许多其他保护隐私的技术，包括基于区块链的 P^{2}B-Trace [18] , 以及利用密码学原语的 BeeTrace [17] 和 Epione [26]。与这些工作使用相对位置不同，我们的方案使用绝对位置，因此除了能够追踪直接接触者外，还能追踪间接接触者（通过受污染环境传播）。

在与接触者追踪无关的领域，近期有许多将差分隐私与密码学技术相结合的研究 [27]。安全洗牌和基于加密的技术 [9, 20] 有助于减少差分隐私引入的误差，而差分隐私则有助于实现更高效的安全计算。我们的方案 ContactGuard 受到这一系列思想的启发，并利用 Geo-I 加速多方计算。此外，许多基于位置的服务（LBS）应用中也存在隐私问题 [23, 25, 22, 11]。

6 结论

在这项工作中，我们研究了隐私保护接触者追踪（PPCT）问题，旨在识别患者的密切接触者，同时保护患者和用户的位置隐私。为解决这一问题，我们提出了一种名为 Contact-Guard 的准确高效方案，该方案借助 Geo-不可区分性（Geo-I）加速安全多方计算（MPC）。实验结果表明，与 MPC 基线相比，ContactGuard 实现了显著的加速比，同时保持了出色的有效性（以召回率和精确率衡量）。未来，我们将探索更先进的数据压缩方法 [16, 15]，以实现更优的隐私与效用权衡。

参考文献

[1] Miguel E. Andres、Nicolas E. Bordenabe、Konstatinos Cjhatzikokolakis 和 Catuscia Palamidessi。《Geo-indistinguishability：基于位置的系统的位置差分隐私》。

发表于 *CCS*,
2013 年。

[2] Eunjoon Cho、Seth A. Myers 和 Jure Leskovec。友谊与移动性：基于位置的社交网络中的用户移动。《KDD》，2011。[3] Graham Cormode、Tejas Kulkarni 和 Divesh Srivastava。在本地差分隐私下回答范围查询。《VLDB》，2019。[4] Teddy Cunningham、Graham Cormode、Hakan Ferhatosmanoglu 和 Divesh Srivastava。基于本地差分隐私的真实世界轨迹共享。《VLDB》，2021。[5] Yanan Da、Ritesh Ahuja、Li Xiong 和 Cyrus Shahabi。REACT：通过隐私增强型移动追踪实现实时接触者追踪与风险监测。《ICDE》，2021。[6] Josep Domingo-Ferrer、David S´anchez 和 Alberto Blanco-Justicia。差分隐私的局限（及其在数据发布和机器学习中的误用）。《ACM通讯》，2021。[7] Bowen Du、Yongxin Tong、Zimu Zhou、Qian Tao 和 Wenjun Zhou。面向需求的电动汽车共享充电桩规划。《KDD》，第1330–1338页，2018。[8] John C Duchi、Michael I Jordan 和 Martin J Wainwright。本地隐私与统计极小极大速率。《FOCS》，2013。[9] Ulfar Erlingsson、Vitaly Feldman、Ilya Mironov、Ananth Raghunathan、Kunal Talwar 和 Abhradeep Thakurta。洗牌放大：通过匿名性实现从本地差分隐私到中心差分隐私的转换。《SODA》，2019。[10] Luca Ferretti、Chris Wymant、Michelle Kendall、Lele Zhao、Anel Nurtay、Lucie Abeler- D¨orner、Michael Parker、David Bonsall 和 Christophe Fraser。对SARS-CoV-2传播的量化表明，数字接触者追踪有助于控制疫情。《科学》，368(6491)，2020。[11] Dawei Gao、Yongxin Tong、Jieying She、Tianshu Song、Lei Chen 和 Ke Xu。空间众包中的Top-k团队推荐。《WAIM》，2016。[12] Wei-jie Guan、Zheng-yi Ni、Yu Hu 等。中国2019年冠状病毒病的临床特征。《新英格兰医学杂志》，382(18):1708–1720，2020。[13] Fumiyuki Kato、Yang Cao 和 Masatoshi Yoshikawa。利用可信硬件实现安全高效的基于轨迹的接触者追踪。《IEEE Big Data》，2020。[14] Dyani Lewis。为何许多国家的COVID接触者追踪工作未能奏效，而一些国家却取得了成功。《自然》，第384–387页，2020。[15] Qiyu Liu、Yanyan Shen 和 Lei Chen。Lhist：面向海量空间数据的多维直方图学习。《ICDE》，2021。[16] Qiyu Liu、Yanyan Shen 和 Lei Chen。HAP：一种基于增强鸽巢原理的高效汉明空间索引。《SIGMOD》，2022。[17] Xiaoyuan Liu、Ni Trieu、Evgenios M. Kornaropoulos 和 Dawn Song。BeeTrace：打破数据孤岛的安全接触者追踪统一平台。《IEEE Data Eng. Bull.》，2020。

[18] Zhe Peng、Cheng Xu、Haixin Wang、Jinbin Huang、Jianliang Xu 和 Xiaowen Chu。Pb- trace：用于应对大流行病的隐私保护型区块链接触者追踪。《SIG- MOD》，2021。[19] Pablo Rodr´ıguez、Santiago Gra˜ña、Eva Elisa Alvarez-Le´on、Manuela Battaglini、Fran- cisco Javier Darias、Miguel A Hern´an、Raquel L´opez、Paloma Llaneza、Maria Cristina Mart´ín、Oriana Ramirez-Rubio 等。评估数字接触者追踪流行病学影响的基于人群的对照实验。《自然·通讯》，12(1):1–6，2021。[20] Amrita Roy Chowdhury、Chenghong Wang、Xi He、Ashwin Machanavajjhala 和 Somesh Jha。CrypT：不可信服务器上的密码学辅助差分隐私。《SIGMOD》，2020。[21] Marcel Salath´e、Christian L Althaus、Nanina Anderegg、Daniele Antonioli、Tala Ballouz、Edouard Bugnion、Srjan Capkun、Dennis Jackson、Sang-Il Kim、James Larus 等。瑞士数字接触者追踪对SARS-CoV-2有效性的早期证据。medRxiv，2020。[22] Jieying She、Yongxin Tong、Lei Chen 和 Tianshu Song。考虑反馈的社交活动参与者安排。《SIGMOD》，2017。[23] Qian Tao、Yongxin Tong、Zimu Zhou、Yexuan Shi、Lei Chen 和 Ke Xu。空间众包中的差分隐私在线任务分配：一种基于树的方法。《ICDE》，2020。[24] Yongxin Tong、Ye Yuan、Yurong Cheng、Lei Chen 和 Guoren Wang。时空众包数据管理技术综述。《软件学报》，28(1):35–58，2017。[25] Yongxin Tong、Yuxiang Zeng、Bolin Ding、Libin Wang 和 Lei Chen。空间众包中的双边在线微任务分配。《TKDE》，2021。[26] Ni Trieu、Kareem Shehata、Prateek Saxena、Reza Shokri 和 Dawn Song。Epione：兼具强隐私保护的轻量级接触者追踪。《IEEE Data Eng. Bull.》，2020。[27] Sameer Wagh、Xi He、Ashwin Machanavajjhala 和 Prateek Mittal。DP-cryptography：在新兴应用中融合差分隐私与密码学。《ACM通讯》，64(2):84–93，2021。[28] Ryan Woo 和 Engen Tham。一座中国城市称，在大型中心为3万人进行新冠肺炎检测，并抓获逃跑者。路透社，2021。[29] Samee Zahur 和 David Evans。Obliv-C：一种用于可扩展数据无关计算的语言。《IACR Cryptol. ePrint Arch.》，2015:1153，2015。

算法 2： Geo-I[1]

输入： ϵ，$l = (x,\ y)$。

输出： l'。**1** 在 $[0, 2\pi)$ 中均匀抽取 θ **2** 在 $[0, 1)$ 中均匀抽取 p **3** $d = -\frac{1}{\epsilon}(W_{-1}(\frac{p-1}{e}) + 1)$ **4** $x' = l.x + \cos(\theta) \cdot d$ **5** $y' = l.y + \sin(\theta) \cdot d$ **6 返回** $l' = (x',\ y')$

表 2：本文使用的主要符号说明。

符号	说明
u, U	一个用户及 n 用户集合
p, P	一名患者和患者集合 m
l, l'	一个位置及其扰动位置
L_u, L'_u	用户 u 的轨迹及其扰动位置
L_P	所有患者的轨迹集合
$I, \tilde{I}$	高风险地点的索引和噪声索引
r	确定密切接触者的距离阈值
r'	用于确定高风险位 cation
ϵ, ϵ_P	用户和患者的隐私预算

A 其他技术细节

为便于查阅，本文使用的主要符号说明汇总于表 2。

A.1 应用设置与对手模型

A.1.1 应用设置

我们将接触者追踪应用建模为客户端-服务器模型（如图 1a 简要所示）：

客户端。 每位用户 $u \in U$ 对应一个客户端。每位用户将自己的位置L_u 存储在移动设备本地。

服务器端。 患者的所有轨迹（如定义4中的 L_P 所示）都会汇总并存储在服务器上。在我们的接触者追踪应用中，服务器由政府所有，具体而言，是由中央公共卫生组织（例如疾病控制与预防中心）所有。这一设定符合现实情况：政府通常会收集确诊患者的行踪，以尽量减少进一步传播，通常会先追踪患者的密切接触者。例如，香港[4]的法律对此作出了规定。

为了进行接触者追踪，无需存储每位患者的单独轨迹，只需存储所有患者轨迹的并集（表示为 L_P）即可。这是因为，对于给定用户，我们的目标仅仅是确定该用户是否与某位患者 $p \in P$曾到访同一附近位置，无需具体识别是哪位患者。这在一定程度上增强了隐私保护，因为不会识别和存储任何单个患者的轨迹。

总而言之，我们的应用中有两个主要角色：用户（客户端）和政府（服务器）。

[4]https://www.elegislation.gov.hk/hk/cap599D!en?索引 CS=N

表3：基线方法与 ContactGuard 的比较

方法	MPC	Geo-I	ContactGuard
准确率	√	×	√
效率	×	√	√

对于服务器上的对手而言，它会尽可能从用户那里获取更多私人信息（获取有关位置 L_{u_i} 的信息）。在此模型下，我们提出的解决方案需要限制从用户共享到服务器的信息。当需要有关 L_u 的信息时，计算过程需要受到隐私保障，关于， L_u。

同样，对手也可能存在于客户端。也就是说，对手会竭尽全力从服务器（存储患者位置 L_P）或其他用户（获取有关其他用户 L_u 的信息）处尽可能多地获取私人信息。因此，我们提出的解决方案也需要限制服务器向每个用户共享的信息。当需要获取有关 L_P 的信息时，计算过程需要受到隐私保障的保护，关于， L_P。

A.1.2 性能指标

考虑到我们的具体应用——接触者追踪，我们根据以下标准衡量解决方案的质量：

(1) 有效性。我们使用召回率、精确率和准确率作为衡量潜在解决方案有效性（效用）的关键指标。在接触者追踪应用中，召回率定义为所有接触者中被正确识别的接触者所占的比例。精确率定义为解决方案识别出的所有接触者中被正确识别的接触者所占的比例。准确率定义为正确分类的实例（包括正确识别的接触者和非接触者）占所有用户的比例。

虽然这三项指标都至关重要，但在接触者追踪应用中，召回率被认为相对更重要，因为它衡量了解决方案消除假阴性（尽可能识别出所有真实接触者）的能力。在疫情控制应用中，这一点至关重要，因为漏掉真实接触者可能会导致疾病进一步失控传播。以中国的新冠肺炎疫情控制为例，当某个区发现数例病例时，全市所有市民都需要接受强制检测，以尽可能降低漏掉阳性病例的可能性 [28]。

（2）效率。我们使用端到端运行时间作为衡量所提出解决方案效率的指标。端到端运行时间是处理 U 中所有用户、以确定他们是否为接触者所需的时间。由于接触者追踪应用通常每天运行一次，如果某个解决方案无法在 24小时内处理所有用户，就会被认为效率不够，因而不切实际。

A.2 基线方案

表3列出了两种基线方法的比较。

算法3给出了 MPC 基线的详细步骤。我们仅展示由服务器（患者）和客户端（用户）共享并调用的主要流程。该流程的输入包括一个类型为 ProtocolIO 的对象 io 。服务器端的输入对象io 包含所有患者的轨迹 L_P 。客户端的输入对象 io 保存了

算法3： MPC位置比较

输入： r，δ，ProtocolIO io。

输出： 真/假。

```
1  n1 :=ocBroadcast(io.n, 参与方=1)
2  n2 :=ocBroadcast(io.n, 参与方=2)
3  保密浮点数组1 =ocToOblivFloatArray(io.x, 参与方=1)
4  保密浮点 ox 数组2 =ocToOblivFloatArray(io.x, 参与方=2)
5  保密浮点 oy 数组1 = ocToOblivFloatArray(io.y, 参与方=1)
6  保密浮点 oy 数组2 = ocToOblivFloatArray(io.y, 参与方=2)
7  保密整数 time 数组1 = ocToOblivIntArray(io.time, 参与方=1)
8  保密整数 time 数组2 = ocToOblivIntArray(io.time, 参与方=2)
9  对每个 i ∈ [0...n1 − 1] 执行
10     对 j ∈ [0...n2 − 1] 执行
11         obliv 浮点数 d x :=ox 数组1[i] - ox_数组2[j]
12         保密浮点数 d y :=oy 数组1[i] - oy_数组2[j]
13         obliv float d 平方: = d x * d x + d y*d y
14         obliv int t1 :=时间 array1[i]
15         obliv int t2 :=time array2[j]
16         不经意整数时间差: = 时间差（t1, t2）
17         不经意 如果 （d 平方 <= r * r 且时间差 <= δ）则
18             返回真
19 返回假
```

用户 u的轨迹 L_u。算法3使用安全计算处理私有输入 io，以确定用户 u （客户端）是否为接触者。

从第3行到第8行，调用 Obliv-C 中的例程 ocToOblivFloatArray，将私有输入（浮点型数组）转换为安全计算期间使用的不经意数据类型。同样，例程 ocToOblivIntArray 用于将整数数组转换为不经意数据类型。在我们的实现中，我们将所有访问位置的 x 和 y 坐标存储在浮点数组中（io.x 和 io.y），并将时间戳存储为整数数组。

请注意，调用例程 ocToOblivFloatArray 时，会传入参数 party=1，以从患者处获取私有输入（服务器，即我们所称的第一方）。传入 party=2 时，该例程会从用户处获取私有输入（即我们所称的第二方）。私有输入转换为不经意数据类型后，所有基于这些不经意数据类型进行的计算也都能保证是不经意的。

访问位置数量 $|L|$ 存储在 io.n 中（一个整数）。由于我们不将访问位置数量视为私有输入，因此会将其广播，并存储为 n_1 （$|L_P|$ 对于患者）和 n_2 （$|L_u|$ 对于用户）。

然后，我们遍历患者的所有位置以及用户的所有位置，检查访问位置与时间戳对是否满足定义5中接触定义的时空约束。所有计算都是无感知的，包括计算两个位置之间欧几里得距离的平方，以及计算两个时间戳之间以秒为单位的时间差，因为这些计算基于无感知输入（例如，由私有输入转换而来的 ox array1），且操作仅涉及底层 Obliv-C 提供的无感知求和与乘法。

在最后一步（第17行），将欧几里得距离的平方与 $r*r$ 进行比较，$r*r$ 是距离阈值 r 的平方。此外，将以秒为单位的时间差与 δ 进行比较，δ 是时间差阈值。 r 和 δ 是我们 PPCT问题的输入，如定义6所述。此比较使用 Obliv-C 提供的安全原语 obliv if，因此程序的执行（程序计数器或运行时间）不会泄露有关底层私有输入（无感知变量）的任何信息。无论条件的值如何，都会执行*obliv if* 的主体。

时间复杂度。 从第3行到第8行，将私有输入转换为不经意数据类型所需的时间为 $\mathcal{O}(n_1 + n_2)$。主要开销来自遍历患者的所有位置以及用户的所有位置，所需时间为 $\mathcal{O}(n_1 * n_2) \rightarrow \mathcal{O}(|L_P||L_u|)$。总体而言，对于每个用户，MPC 过程算法3所需时间为$\mathcal{O}(|L_P| + |L_u| + |L_P||L_u|) \rightarrow \mathcal{O}(|L_P||L_u|)$。对于服务器 S处理完所有用户，所需时间为 $\mathcal{O}(\sum_u |L_u||L_P|) \rightarrow \mathcal{O}(|U||L_P| \max_u |L_u|)$。

请注意，此处的时间复杂度分析假设安全原语（例如，ocTo OblivFloat、无感知求和与乘法、无感知条件判断）均以常数时间（$\mathcal{O}(1)$）运行。实际中，安全原语的运行速度比非安全语句慢多个数量级。我们采用较高层次的分析，考察安全原语的总执行次数，而不评估 Obliv-C 所生成的底层安全电路的门数。在我们提出的方法 ContactGuard 中，我们通过大幅减少所使用的安全原语数量来优化运行时间。

隐私分析。

定理3。 *算法3满足定义6中的隐私要求。*

证明。 由于我们对私有输入使用 MPC（由 Obliv-C 提供）操作，因此显然能够保证隐私。我们证明，除访问位置集合的大小外，算法3不会泄露有关 L_u 用户和 L_P 患者的任何信息。

关于 L_u：从第3行到第8行，均转换为不经意数据类型。从第9行到第18行的迭代中的计算仅使用无感知变量（例如，ox 数组2）。因此， L_u 得到了严格保护。请注意，在我们的问题设定（定义6）中，轨迹长度（$|L_u|$）不被视为敏感信息；事实上，我们会在第2行广播 $|L_u|$ ，且该信息对双方公开可见。

关于 L_P：从第3行到第8行，均转换为不经意数据类型。从第9行到第18行的迭代中的计算仅使用无感知变量（例如，ox 数组1和 time 数组1）。因此， L_P 得到了严格保护。

□

A.3 ContactGurad

该方法包含以下步骤。

步骤 1. 位置扰动：每位用户对自己的访问位置集合 L_u 进行扰动，得到扰动后的访问位置集合 L'_u。用户将 L'_u 提交给服务器。

步骤 2. 子集选择：服务器将患者的轨迹 L'_u 与 L_P进行比较。服务器会记录与患者的某个到访地点距离 r' 以内的高风险位置的索引（例如，第 3/4 个位置）。随后，服务器将已识别的高风险位置的索引列表（记为$\tilde{I}$，并经过隐私保护）返回给用户。

步骤 3. 加速 MPC：用户根据以下内容选择访问位置的子集 $\tilde{I}$，由服务器在步骤 2中返回，然后使用多方计算协议，将该位置子集与服务器上的患者轨迹 L_P 进行比较。如果子集中存在一个位置

（以及对应的访问时间戳）符合定义5中接触者的定义，用户 u将被识别为接触者。否则，该用户不会被识别为接触者。

A.3.1 证明

根据组合机制 $K_{1,\ 2}$(l) $=$ (K_1(l_1) K_2(l_2)) 的定义，式（2）成立。该机制独立地将机制 K_1 应用于第一个位置 l_1，并将机制 K_2 应用于第二个位置 l_2。因此，生成 o_1 和 o_2 的概率，分别是通过 K_1 生成 o_1 的概率与通过 K_2 生成 o_2 的概率的乘积。

式（4）成立，是因为 K_1 满足 ϵ_1-Geo-I（[1]），且 K_2 满足 ϵ_2-Geo-I。根据 [1], 中 Geo-I 的定义，对于两个输入位置 l_1 和 l_1'，生成相同输出位置 o_1 的概率受限于 $e^{\epsilon_1 \cdot d_{(l_1,\ l_1')}}$。 [1] 中两个分布 d_ρ(M(x) M(x')) 之间的乘法距离定义为 $|\ln\ ^{M_{(x)}}{}_{M(x')}|$（见原始论文 [1]），这里我们将 ln 函数变换回指数形式。

根据 d_∞(l, d_∞(l, l') $= max_i d$(l_i, l_i') 的定义可得公式 (5)。 d(l_1, l_1') $\leq max_i d$(l_i, l_i') 和 d (l_2, l_2') $\leq max_i d$(l_i, l_i') 均成立。

隐私分　析。本步骤的隐私保障由以下定理给出。

证明。不失一般性，我们聚焦于 L_P 中的一个特定位置 l。假设 l 与 L_u' 中第 i 个位置的距离在 r' 以内。那么构造出的精确索引为i $= \{0, \ldots, 1, \ldots, 0\}$，其中第 i 个位置的值为 1，其余所有位置的值均为 0。然后，对于给定的输出 o $= \{0, \ldots, 1, \ldots, 0\}$，使用随机响应机制时，在第 i 个位置生成该输出的概率为 e^{ϵ_P}/($e^{\epsilon_P}+1$)（忽略其他位置的概率，因为对于相邻输入 L_P'，这些概率相同，因此会相互抵消）。

对于相邻输入 L_P'，其中除 $l' \in L_P'$ 外的所有位置都相同，且 l' 位于 L_u' 中第 i 个位置以及 L_u' 中所有其他位置的距离 r' 之外。为此相邻输入构造的精确索引为 i $' = \{0, \ldots, 0, \ldots, 0\}$，其中第 i 个位置的值为 0。于是，在第 i 个位置生成相同输出 $\mathbf{0} = \{0, \ldots, 1, \ldots, 0\}$ 的概率为 $1/(e^{\epsilon_P}+1)$（同样忽略其他位置的概率）。因此，两个概率之比为:

$$\frac{\Pr[R(\mathbf{i}) = \mathbf{o} | l \in L_P)]}{\Pr[R(\mathbf{i}') = \mathbf{o} | l' \in L_P')]} = \frac{e^{\epsilon_P}/(e^{\epsilon_P}+1)}{1/(e^{\epsilon_P}+1)} = e^{\epsilon_P}. \tag{7}$$

因此，子集选择步骤满足 ϵ_P-本地差分隐私。 □

A.3.2 算法

下面我们通过一个运行示例，更好地说明子集选择的基本思想 .

示例 2. 该示例如图*2b*所示。在此示例中，用户 u 在时间 t_1 到访了位置l_1，并在时间 t_2到访了 l_2。在步骤 *1*（位置扰动）中，用户将位置 l_1 扰动为l_1'，并将 l_2 扰动为l_1'。集合 L_u' 由这两个扰动位置构成，并被发送到服务器。

在此示例中，患者于时间 t_3 到访地点 l_3。随后，服务器将包含 l_1' 和 l_2' 的 L_u' 与 l_3 进行比较。由于 l_2' 与 l_3 的距离足够近（在距离 r' 以内），l_2' 被识别为高风险位置（并被选中纳入子集），而 l_1' 未被选中。然后，服务器将高风险位置子集的索引（记作 $I = \{2\}$）返回给用户，表示第 *2* 个位置是高风险位置。

算法4： 子集选择

输入： L'_u，L_P，r'，ϵ_P。

输出： $\tilde{I}$.

$I := \{\}$
$i := 0$
$\gamma := e^{\epsilon_P}/(e^{\epsilon_P}+1)$
对每个 $l' \in L'_u$ **do**
  $i = i + 1$
  标志：$= 0$
  对每个 $l \in L_P$ **执行**
    如果 $d(l', l) \leq r'$ **则**
      标志 $= 1$
  样本 $b :=$ 伯努利分布(γ)　　// 随机响应
  如果b *= 1* **则**
    标志 = 1- 标志　　// 翻转结果
  如果标志 *= 1* **则**
    $\tilde{I}$.插入(i)
返回 $\tilde{I}$

出于隐私考虑，服务器在返回确切的高风险索引之前，会使用随机响应对其进行扰动。随后，服务器将噪声索引返回给用户。在此示例中，$I = \{2\}$ 也可以表示为 i $= \{0, 1\}$，表示第 *1* 个位置不是高风险位置（用值 *0* 表示），而第 *2* 个位置是高风险位置（用值 *1* 表示）。每一位都会通过随机响应进行扰动。给定隐私预算 ϵ_P，以概率 $e^{\epsilon_P}/(e^{\epsilon_P}+1)$ 返回原始值；否则，以概率 $1/(e^{\epsilon_P}+1)$ 返回翻转后的值（*0* 变为 *1*，*1* 变为 *0*）。

算法4详细介绍了如何选择高风险位置子集。算法首先将返回的索引集合初始化为空集，然后针对每个扰动位置 l'，检查它是否与患者到访的某个位置 l 的距离在 r' 以内（这些位置存储在 L_P中）。对于每个位置，我们使用随机响应将原始值（变量 fl ag）转换为噪声值。最后，返回噪声索引 I 给用户 u。

请注意，参数 r' 是一个可调变量，用于平衡 ContactGuard 的准确性与效率权衡。如果 r' 设为较大值，那么所有位置都会被纳入高风险位置子集，并在后续的多方计算中使用，因此会产生较高的开销。如果 r' 设为较小值，那么扰动位置必须接近患者到访过的某个位置，才能被纳入高风险位置子集，这会导致更多假阴性。

算法5描述了如何仅使用位置子集，为多方计算协议准备私有输入。

A.3.3 时间复杂度

我们对 ContactGuard 的每个步骤进行时间复杂度分析。

步骤 1，位置扰动：客户端（用户）运行算法 1。时间复杂度为$\mathcal{O}(|L_u|)$，与输入 L_u中给定的访问位置数量成线性关系。

算法 5： 准备 MPC 输入

输入： L_u,$\tilde{I}$.
输出： ProtocolIO io。

$io :=$ 初始化一个 ProtocolIO 对象
$io.n = \tilde{I}$.大小()
$i := 0$
遍历 $l_{idx} \in \tilde{I}$ **do**
    $io.x[i] = L_u[l_{idx}].x$
    $io.y[i] = L_u[l_{idx}].y$
    $io.time[i] = L_u[l_{idx}].time$
    $i = i + 1$
返回 io

步骤 2，子集选择：服务器端执行子集选择。时间复杂度为 $\mathcal{O}(|L'_u||L_P|) \rightarrow \mathcal{O}(|L_u||L_P|)$，因为它会将 L'_u 中的每个位置与 L_P中的每个位置进行比较。

步骤 3，加速 MPC：令 $|I|$ 为步骤 2 中选出的子集大小。将私有输入转换为不经意数据类型的不经意变换需要 $\mathcal{O}(|n_I| + |L_P|)$ 才能完成。主要开销来自遍历患者的所有位置，以及用户访问位置子集中的位置，这需要 $\mathcal{O}(|L_P||I|)$ 才能完成。总体而言，对于每位用户，多方安全计算过程需要 $\mathcal{O}(|L_P| + |I| + |L_P||I|) \rightarrow \mathcal{O}(|L_P||I|)$。

总体而言，服务器处理完所有用户需要 $\mathcal{O}(\sum_u |I_u||L_P|) \rightarrow \mathcal{O}(|U||L_P|\max_u |I_u|)$，其中 I_u 表示每位用户的子集 u。时间复杂度以一次 MPC 操作为一个基本操作，并且超过了步骤 2 中子集选择所需的 $\mathcal{O}(|L_u||L_P|)$ 时间；该步骤不需要安全计算。

A.3.4 隐私分析

我们展示 ContactGuard 的端到端隐私分析。

步骤 1，位置扰动： L_u 通过定理 1 中所示的 ϵ-General-Geo-I 得到保护。客户端不了解患者位置（L_P），因此步骤 1 对于 L_u 和 L_P均具有隐私性。

步骤 2，子集选择：计算仅针对扰动位置 L'_u进行。根据 Geo-I 的后处理性质，计算过程对于L_u仍具有隐私性。关于患者位置 L_P，定理 2 表明，返回的噪声索引满足ϵ_P-局部差分隐私，针对 L_P。因此，该步骤对于 L_u和 L_P均具有隐私性。

步骤 3，加速 MPC：隐私性由基于私有输入、通过 MPC 过程（由 Obliv-C 提供）执行的操作来保证。在客户端，从L_P 中选择 L_u 的子集，并使用噪声索引 I，随后将其转换为不经意数据类型。在服务器端， L_P 也会被转换为不经意数据类型。所有后续计算均基于无痕算子。

B 补充实验结果

Gowalla 数据集：这是一个基于位置的社交网络签到数据集 [2], , 记录了用户到访位置的纬度和经度。该数据集包含不同

表 4：Gowalla 数据集上不同方法的召回率/精确率/准确率与运行时间之间的权衡。$|U| = 200,\ \epsilon = 4.0.$

方法	Geo-I	MPC	ContactGuard
召回率	70.0%	100%	**100%**
精确率	46.7%	100%	**100%**
F_1 分数	56.0%	100%	**100%**
准确率	76.8%	100%	**100%**
运行时间（秒）	0.08	99.35	**39.31（速度快 2.53 倍）**

表 5：Gowalla 数据集上不同方法在召回率/精确率/准确率与运行时间之间的权衡。$|U| = 400,\ \epsilon = 3.0.$

方法	Geo-I	MPC	ContactGuard
召回率	39.0%	100%	**88.89%**
精确率	42.0%	100%	**100%**
F_1 分数	35.0%	100%	**94.1%**
准确率	80.5%	100%	**99.75%**
运行时间（秒）	0.39	194.99	**77.03（快 2.52 倍）**

城市中的记录。我们选取旧金山市的记录，将 GIS 坐标映射到范围为 $[0, 10549] \times [0, 8499]$ 的 X、Y 笛卡尔坐标，以便将 Geo-I 直接应用于 X、Y 坐标。按照文献 [7], 中的方法，我们随机选取一小部分用户作为患者（患者人数设为 2 至 8 人，患者与用户的比例约为 1%），其余用户则作为待测试用户。

合成数据集：我们生成的位置均使用 X、Y 坐标，其取值范围为 $[0, 10549] \times [0, 8499],$，与 Gowalla 数据集的取值范围相同。

B.1 系统配置

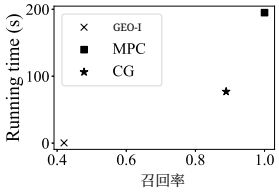
实验在 CentOS Linux 系统上进行，使用 Intel(R) Xeon(R) Gold 6240R CPU @2.40GHz 和 1007G 内存。我们使用 C 实现这些方法并进行实验。MPC 扩展由 Obliv-C [29] 提供。我们在同一台机器上运行两个进程来模拟客户端-服务器 MPC 协议，并忽略通信时间。

详细结果分别见表 4 和表 5。

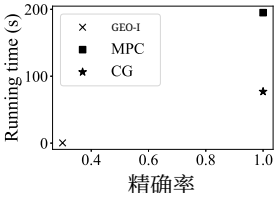
与 Geo-I 基线相比，CG 方法在召回率、精确率和准确率方面分别提升了 2.28×、2.38× 和 1.24× 。另一方面，与精确 MPC 基线相比，CG 方法的速度约快 2.5× 倍（表 4 和表 5 中均如此）。

在另一种隐私预算较小的设置下（如图 7 所示，$|U| = 400, \epsilon = 3.0$），共轭梯度法仍能达到相当高的有效性。除了保持 100% 的精确率外，其召回率为 88.89%，准确率为 99.75%。

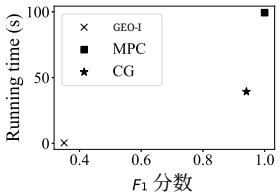
图 8 展示了针对合成数据集，在改变用户数量（$|U|$）时各项有效性指标（召回率、精确率、F_1 和准确率）。



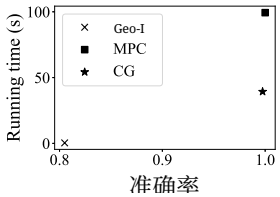
(a) 召回率与运行时间。



(b) 精确率与 Running 时间。

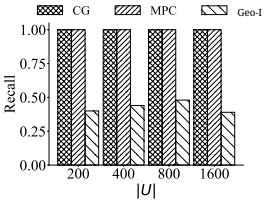


(c) F_1 分数与 Running 时间。

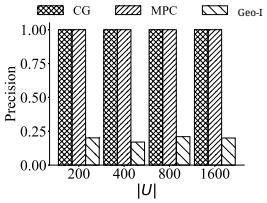


(d) 准确率与运行时间。

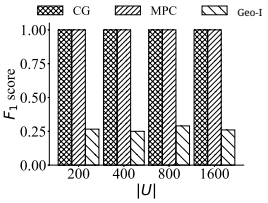
图 7：Gowalla 数据集上不同方法的有效性（召回率/精确率/F_1/准确率）与运行时间之间的权衡。 $|U| = 400,\ \epsilon = 3.0.$



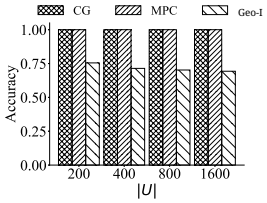
(a) 召回率。



(b) 精确率。



(c) F_1 分数。



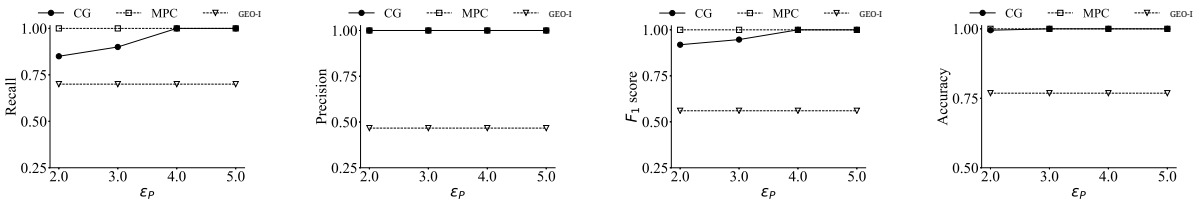
(d) 准确率。

图 8：在合成数据集上改变用户数量（$|U|$）时的有效性（召回率/精确率/F_1/准确率）。 $\epsilon = 4.0.$

B.2 Gowalla 数据集上的更多结果

关于改变 ϵ_P 的更多内容：

当改变 ϵ_P 时，CG 方法在召回率/精确率/F_1/准确率方面表现出较高的有效性。尽管当 $\epsilon_P = 2.0$ 时，召回率降至约 0.8，但在给定较大值时，其表现与 MPC 相近。



(a) 召回率。 (b) 精确率。 （c） F_1 得分。 （d）准确率。

图9：在 Gowalla 数据集上改变患者（ϵ_P）的隐私预算时的有效性（召回率/精确率/F_1/准确率）。 $eps = 4.0,\ \ |U| = 400.$