分类号 TP957 学号 0123456

UDC 密级 公开

专业学位硕士学位论文

云环境下隐私保护聚类方法关键技术研究

硕士生姓名 邓晏湘

专业学位类别 计算机技术

专业学位领域 隐私保护聚类

指导教师 付绍静 教授

协助指导教师 柳林 讲师

国防科技大学计算机学院

二〇二三年二月

Research on Key Technologies of Privacy-preserving Clustering Method in Cloud Environment

Candidate: **DengYanxiang**

Supervisor: **Prof. FuShaoJing**

Associate Supervisor: **Lecture. LiuLin**

A thesis

Submitted in partial fulfillment of the requirements

for the professional degree of **Master of Engineering**

in **Computer Technology**

Graduate School of National University of Defense Technology

Changsha, Hunan, P. R. China

February, 2023

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科技大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：云环境下隐私保护聚类方法关键技术研究

学位论文作者签名：_____ 日期： 年 月 日

学位论文版权使用授权书

本人完全了解国防科技大学有关保留、使用学位论文的规定。本人授权国防科技大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅;可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密学位论文在解密后适用本授权书。）

学位论文题目：云环境下隐私保护聚类方法关键技术研究

学位论文作者签名：_____ 日期： 年 月 日

作者指导教师签名：_____ 日期： 年 月 日

目 录

表 目 录

图 目 录

摘　要

国防科学技术大学是一所直属中央军委的综合性大学。1984 年, 学校经国务院、中央军委和教育部批准首批成立研究生院，肩负着为全军培养高级科学和工程技术人才与指挥人才，培训高级领导干部，从事先进武器装备和国防关键技术研究的重要任务。国防科技大学是全国重点大学，也是全国首批进入国家“211 工程”建设并获中央专项经费支持的全国重点院校之一。学校前身是 1953 年创建于哈尔滨的中国人民解放军军事工程学院，简称“哈军工”。

关键词：国防科学技术大学; 211; **哈军工**

Abstract

National University of Defense Technology is a comprehensive national key university based in Changsha, Hunan Province, China. It is under the dual supervision of the Ministry of National Defense and the Ministry of Education, designated for Project 211 and Project 985, the two national plans for facilitating the development of Chinese higher education.

NUDT was originally founded in 1953 as the Military Academy of Engineering in Harbin of Heilongjiang Province. In 1970 the Academy of Engineering moved southwards to Changsha and was renamed Changsha Institute of Technology.The Institute changed its name to National University of Defense Technology in 1978.

Key Words：NUDT; MND; ME

符号使用说明

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物，N- 苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物，N- 苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度

第一章　绪论

1.1　研究背景与意义

1.2　聚类综述

1.3　研究内容和创新点

1.4　论文的组织结构

第二章　相关研究综述

2.1　相关理论综述

2.2　国内外研究现状

第三章　隐私保护 k-means 聚类方法研究

3.1　引言

随着现代社会数字化的不断演进，数据使用量呈指数级增加，个人和小型组织越来越难以在内部计算机服务器上维护所有重要信息、运行大型程序和系统。为解决这些问题，云计算应运而生。自云计算在 2006 年被提出后，它就被认为是能够推动下一代互联网革命的技术，并迅速成为了研究领域的热门话题[1]。近年来，随着研究的发展和设备的进步，机器学习从学术研究落地到生活的方方面面，但是大多数机器学习任务对于设备的性能要求较高，需要存储海量的数据才能取得较好的结果。大型公司有能力承担设备费用，利用机器学习的便利开展各种各样的服务，但是资源受限的小公司和个人的需求常常难以被满足。

为解决上述问题，云计算场景下出现了一种新的服务类型 -MLaaS，即“机器学习即服务（Machine Learning as a Service）”，它使得技术可以扩展，并且价格合理[2]。在 MLaaS 中，海量的数据需要被上传到云计算中心，这一过程也被称之为外包计算。由于用户在数据上传后失去了对数据的完全控制，因此会更加关心隐私安全问题。同时云计算服务模型的复杂性、实时性，数据的多元异构特点以及终端资源有限，传统的数据安全隐私和隐私保护方法无法直接用来保护云计算中的大量数据[3]。

聚类（Clustering）是一种非常流行的无监督机器学习技术，它了能够将相似的输入元素划分到同一个簇（cluster）中。聚类的应用领域非常广泛，从业务分析到医疗保健等诸多领域。在许多这些应用场景中，敏感信息在被正确聚类的同时，也不应该被泄漏。此外，现在经常需要将不同来源的数据组合起来进行训练以提升分析质量，庞大的数据量对资源受限的用户带来巨大的压力，因此，通常需要将复杂的计算外包给强大的云服务器进行处理，这就要求我们设计有效隐私保护聚类方案[4]。

目前，为了保护聚类过程中输入的敏感数据的隐私，已经有了许多研究成果，涵盖各个方面。在设计隐私保护聚类协议时，通常考虑两个不同的场景。在多方计算的场景下[5]，两个及两个以上的数据所有者共同执行安全计算协议，除了输出之外的任何内容都不会泄漏给彼此，比较常见的是两方计算。同时，一些研究也会在模型设计中引入半诚实参与方（通常是服务器）来辅助计算。相对应的，另一种场景则是外包计算[6]，一个或多个数据所有者将将计算（或存储）外包给其他参与方，我们假设这些参与方可以为数据所有者进行聚类而无需知道关于输入数据的任何信息。由于外包计算旨在使用外部资源，数据所有者通常不应该参与

协议的执行，处于离线状态，但是这一点通常难以完全实现。值得一提的是，一些多方计算（MPC）议也可以用于外包计算场景，只需要数据所有者在多个不共谋的参与方之间秘密共享自己的数据，然后多个参与方在秘密共享的数据上执行聚类算法。但是，MPC 协议是否支持外包计算很大程度上取决于协议设计，如果数据持有者需要在聚类过程中进行大量的明文计算或者在中间对数据进行解密，那就很难将协议转换到外包计算的场景。

外包计算和多方计算都各有其优势，但是对于资源受限的用户来说，外包计算则更加具有实际意义，基于外包计算的安全计算方法已有诸多研究，但是完全安全的聚类方案通常效率较低，即便是在性能较好的服务器上也需要花费难以接受的时间才能获得最终结果，效率较高的方案通常会牺牲一定的安全性，泄漏中间计算结果，例如簇大小，簇中心，这些都会导致一些隐私安全问题。因此如何设计出安全又高效的外包聚类方案值得深入研究和探索。

针对上述研究现状，我们研究在外包计算场景下，如何在保护用户隐私数据和聚类中间结果以及保证效率可接受的同时，精确的完成聚类计算。首先，我们选择应用最为广泛的 k-means 聚类方案。k-means 聚类是最简单和流行的无监督学习算法之一，能够应用于大型数据集，并且具有良好的收敛性。其次，我们引入 kd-tree 来加速聚类过程，kd-tree 是一种空间划分数据结构，将空间上靠近的数据放在同一个节点中，广泛用于空间相关应用来加速计算。基于此，我们提出了一种基于 kd-tree 的隐私保护 k-means 聚类方法。方案由客户端和双云服务器参与，用户在本地明文基础上构造 kd-tree，然后秘密共享所有数据分别发送给双云，云服务器在密文基础上运行系列安全协议来获取聚类结果后，将密文结果发送给用户，用户进行还原。理论分析与实验验证表明，方案在为用户提供外包聚类计算的同时，保证了数据的安全性，计算的高效性以及聚类结果的精确性。

本章的组织结构如下：第3.2节介绍了安全性定义、基于 kd-tree 的 k-means 聚类以及基于秘密共享的安全多方计算。第3.3节中描述了系统模型、安全模型以及设计目标。第3.4节中提出了一系列基于秘密共享的隐私保护计算模块。第3.5节中提出了基于 kd-tree 的隐私保护 k-means 聚类方案。第3.6从理论上分析了方案的正确性和安全性。第3.7节中对方案进行了全面的实验评估。最后，在3.8节中对本章进行了总结。

3.2 预备知识

3.2.1 安全性定义

3.2.1.1 半诚实模型

半诚实模型（Semi-honest Model），也称为诚实且好奇的模型（Honest-but-curious Model), 指的是参与方会诚实的执行所有协议，但会竭尽所能利用已有的信息获取尽可能多的内容。半诚实的参与方通常是被动的，因为他们除了通过观察执行协议的过程无法采取其他任何行动来获取隐私数据。这样的模型广泛应用于诸多研究来支持密文数据上交互协议的研究。

3.2.2 基于 kd-tree 的 k-means 聚类方法

k-means 聚类是最常用的聚类算法之一，给定簇个数 k，它能够将大小为 n 的数据集划分成 k 个内容不相交的子集。每个簇都有一个中心点，对单个数据而言，该数据点到最终所属簇中心的距离相较于其他簇更短。k-means 聚类算法有多种不同的实现方式，以使聚类更加高效便捷。[7] 论文的核心思想是以 kd-tree 为主要数据结构，设计了一种高效的过滤算法来聚类数据。

3.2.2.1 kd-tree 数据结构

kd-tree 是一种空间划分数据结构，将空间上靠近的点划分到树的同一个节点中[8]，通常应用于加速与点空间相关的计算，例如 k 近邻算法和创建点云。

这里，我们遵循如下规则来创建 kd-tree：

- 在 $k-$ 维数据集中，我们计算数据每个维度的方差，选择方差最大的维度 d_{max} 来划分数据
- 找到 d_{max} 维度数据的中位数 m 作为基准来划分数据集，得到两个子集合
- 对划分出的子集重复上述过程直到 kd-tree 构造完成

kd-tree 数据结构有两个主要的优势，一方面，它能够将可能属于同一个数据集的点划分到同一个树节点中。另一方面，它采取一种高效的方式来将初始空间划分为两个子空间以加速后续数据处理。

3.2.2.2 过滤算法

本小节我们详细介绍[7] 的核心，过滤算法。给定 n 个点，构造的 kd-tree 包含 $O(n)$ 个节点，长度为 $O(\log n)$。对于 kd-tree 中每个节点 u，计算包含的数据个数 $u.count$ 以及加权质心 $u.wgtCent$，即包含点数据之和。中心的计算方式为 $u.wgtCent/u.count$，在构造 kd-tree 的过程中可以连带计算上述内容。初始簇中心采取随机从数据集中选择的方式。

对于 kd-tree 中每个节点，我们维护一个候选簇集合，该集合中的簇均有可能为节点中数据的最终所属簇。对于根节点，候选簇为随机选择的 k 个簇。我们按

照如下方式来选择每个点的候选簇集合：对于每个节点 u，令 C 为包含数据集合，Z 为候选簇集合。首先计算 C 的中心点，然后找到候选簇中距离中心最近的簇 $z^* \in Z$。对比剩余的候选簇 $z \in Z \backslash \{z^*\}$，若 C 中所有数据均距离 z^* 更近，则我们认为 z 不是节点 u 中任何一个数据的最近所属簇，因此我们进行剪枝，即从候选簇中删除 z。若 u 仅剩一个候选簇，即 z^*，则我们认为 z^* 就是 u 中所有数据的所属簇。我们可以将这些数据划分进 z^*，并且将相关的 $u.wgtCent$ 和 $u.count$ 添加到 z^* 中。否则，若 u 为非叶节点，我们对其子节点重复上述过程。若 u 为叶节点，我们计算它到所有候选簇的距离，并分配到最近簇中。

3.2.3 基于秘密共享的安全多方计算

安全多方计算（SMC）首先由[9] 提出，它能够使多个参与方在不泄露自身输入数据的前提下协同进行计算获取结果，广泛用于各种隐私保护方案中。秘密共享是安全多方计算中一个常用工具，最早由 Shamir[10] 和 Blakley[11] 于 1979 年分别提出。秘密共享的基本思路是：将秘密 s 划分为 n 份，分发给 n 个不同的参与方，至少需要 t 个不同的参与者才能够重构秘密，否则失败。接下来，我们以两个参与方 A 与 B 为例，详细介绍秘密共享相关计算。

假设所有数据的范围在环 $\mathbb{Z}_P$ 上，对于 x 的加性秘密共享值 $\langle x \rangle$，有 $\langle x \rangle^A + \langle x \rangle^B = x(\mod P)$。参与方 A 拥有 $\langle x \rangle^A$，参与方 B 同理。重构秘密 x（$Rec(\cdot,\cdot)$），其中一个参与方将分配给自己的秘密发送给另一方，然后计算 $x = \langle x \rangle^A + \langle x \rangle^B (\mod P)$。接下来的说明中，为了简洁我们省略 $\mod P$ 操作。

3.2.3.1 加性秘密共享加法

为计算两个秘密共享值 $\langle x \rangle$（$\langle x \rangle^A, \langle x \rangle^B$）和 $\langle y \rangle$（$\langle y \rangle^A, \langle y \rangle^B$）之和，参与方 A 与 B 分别计算 $\langle z \rangle^A = \langle x \rangle^A + \langle y \rangle^A$ 和 $\langle z \rangle^B = \langle x \rangle^B + \langle y \rangle^B$。对于秘密共享值 $\langle x \rangle$ 与常量 c 的加法，参与方 A 在本地计算 $\langle z \rangle^A = \langle x \rangle + c$，参与方 B 计算 $\langle z \rangle^B = \langle x \rangle^B$。

3.2.4 秘密共享乘法

秘密共享的安全乘法协议首先由 Beaver[12] 提出，为计算乘积 $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$，我们需要一个预计算的乘法三元组 $\langle c \rangle = \langle a \rangle \cdot \langle b \rangle$。其中参与方 i 计算 $\langle e \rangle^i = \langle x \rangle^i - \langle a \rangle^i$ 和 $\langle f \rangle^i = \langle y \rangle^i - \langle b \rangle^i$, 其中 $i \in \{A, B\}$。参与方均运行 $Rec(\cdot,\cdot)$ 来恢复 e 和 f。最后，参与方 A 令 $\langle z \rangle^A = f \cdot \langle a \rangle^A + e \cdot \langle b \rangle^A + \langle c \rangle^A$，参与方 B 令 $\langle z \rangle^B = e \cdot f + f \cdot \langle a \rangle^B + e \cdot \langle b \rangle^B + \langle c \rangle^B$，即计算完成。

值得注意的是，乘法主要分为两个阶段。离线阶段中，预计算的乘法三元组每次进行乘法时都要更新，但是三元组生成的过程是离线的，可以由两方通过不经意传输（Oblivious Transfer）生成[13]，也可以由可信第三方提供[14]。更多关于预

计算乘法三元组的细节可以参考[12]。在线阶段中，两个参与方相互通信以获得最终的计算结果。

3.3 问题描述

3.3.1 系统模型

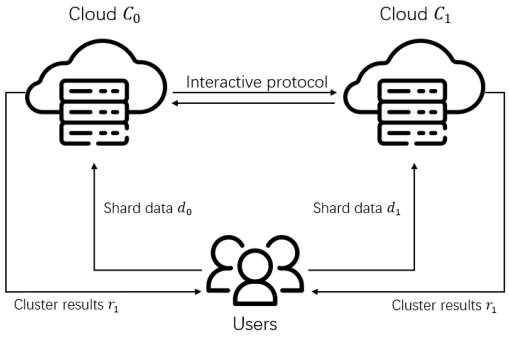

图 3.1 System Model

如图3.1所示，我们的系统由两个部分组成：用户和云服务器。这样的系统模型广泛用于隐私保护研究中[15][16]。这里的客户指的是，持有隐私数据需要聚类来进行数据分析和挖掘的独立用户，他们通常没有足够的计算资源来聚类海量数据，因此需要将计算任务外包以获取结果。常见的实际用户有医疗机构、金融机构等。服务器端通常指的是拥有大量计算资源，提供付费计算服务的云服务器运营厂商例如阿里云、腾讯云等，他们提供机器学习即服务（MLaaS）这一付费模式，用户上传数据后即可离线等待机器学习计算结果。下面我们详细介绍方案中的每一部分：

- **双云服务器**：我们在系统中有两个不共谋的云服务器，标识为 C_0 和 C_1，而且均持有用户秘密共享后的数据。他们会在此基础上执行一系列安全协议来实现隐私保护 k-means 聚类，最后将秘密共享的结果发送给用户。
- **用户**：用户首先在原始数据上构造 kd-tree，然后选择随机数来将数据划分为两份秘密共享值，分别发送给双云服务器。在 k-means 聚类结束后，获取服务器返回的秘密共享结果，运行 $Rec(\cdot,\cdot)$ 来还原最终结果。

3.3.2 安全模型

服务器端与客户端均为半诚实的实体，即二者均会遵循协议内容，不会蓄意停止执行安全协议或是篡改中间计算结果，但会好奇彼此的隐私数据，并尝试从获取的信息中推断其他参与方的信息。虽然云服务器无法还原原始数据，但是可能会企图从计算中间结果中推测数据的分布信息，具体的应对策略为设计安全全

面的计算协议，以防止服务器推断原始信息。

3.3.3 设计目标

本节所述隐私保护外包 k-means 聚类方案设计目标如下：

- **隐私和安全性**：所有外包数据和中间计算结果都不应泄露给双云服务器。同时，服务器无法从秘密共享值推断出原始数据内容以及相关数据特征，例如数据分布。在我们的方案中，上述内容均处于加密状态。
- **高效性**：方案应具备高效性，整个聚类过程主要由双云服务器进行，二者具有丰富的计算和通信资源，能够对海量数据进行处理，极大降低用户计算开销。
- **正确性**：方案应保证计算精度。双云服务器应当能够返回和明文聚类方案一致的计算结果。

3.4 基于秘密共享的隐私保护计算模块

为了使用户与云端能够进行安全的交互，我们基于秘密共享技术设计了一系列基本的计算模块。用户通过产生随机值将明文数据拆分为两份密文发送给双云服务器，两方在秘密共享值的基础上进行系列计算和交互，获取最终结果。

3.4.1 安全欧式距离计算协议

计算簇 z 到点 x 之间的欧式距离的计算公式如下：

$$dist = \sqrt{\sum_{j=1}^{m}(z[j]-x[j])^2} \tag{3.1}$$

其中下标 j 标识数据的第 j 个维度，所有数据均为加性秘密共享值。在所有点均被划分到对应簇后，我们通过计算平均值获得簇的中心点。假设簇 z_i' 包含点个数为 $|z_i'|$，包含的数据为 $x_1, ..., x_{|z_i'|}$，那么簇 z_i' 的中心点计算方式如下：

$$z_i'[j] = \frac{x_1[j] + \cdots + x_{|z_i'|}[j]}{|z_i'|} = \frac{s_i'[j]}{|z_i'|}, 1 \le j \le m \tag{3.2}$$

其中 $s_i'[j]$ 表示簇 z' 中所有点第 j 个维度数据之和。在秘密共享值上进行除法比较困难，因此我们采用文献 ref-wu 中提到的放缩方法来将距离计算中的除法转变为乘法。首先，我们计算全局缩放因子 α 以及簇 z_i 对应的 α_i，计算方式为：

$$\alpha = \prod_{j=1}^{k}|z_j|, \alpha_i = \prod_{j=1 \wedge i \neq j}^{k}|z_j| \tag{3.3}$$

为了方便计算，我们省略公式3.1中的根号计算，将整个公式改写为：

$$dist = \sum_{j=1}^{m}(\alpha x[j] - \alpha_i z_i[j])^2 \tag{3.4}$$

根据秘密共享方案的性质，我们可以针对公式3.4做出改进，简化计算。在一轮迭代中，无论计算什么距离，缩放因子 α 和 α_i 的值以及相关的计算结果都是不变的，因此我们可以在每轮迭代开始计算这些固定值，减少后续冗余计算。同时，参数不相关的计算可以并行进行，减少云服务器交互的次数。

3.4.2 安全比较协议

安全比较场景如下，参与方 p_i 拥有加性秘密共享值 $\langle x_i\rangle$ 和 $\langle y_i\rangle$，其中 $i \in \{0,1\}$，我们希望能够在不泄露 x 和 y 明文值的前提下，获取二者的大小关系 $\delta = LT(x,y), \delta = \langle\delta\rangle_0 + \langle\delta\rangle_1$，其中 $\delta = LT(x,y)$ 的具体含义如下：

$$\mathrm{LT}(x,y) = \begin{cases} 1, & \text{if } x < y \\ 0, & \text{if } x \geq y \end{cases} \tag{3.5}$$

在文献 ref-crypt 中，作者提出了一个高效安全的比较协议来解决百万富翁问题，该方案能够同时比较多对数据，通信开销较小，计算复杂度低。经过多轮不经意传输（oblivious transfer）后，参与方 p_0 和 p_1 分别获得布尔秘密共享结果 $\delta = LT(x,y), \delta = \langle\delta\rangle_0^B \bigoplus \langle\delta\rangle_1^B$。

由于本研究中安全比较的输入与输出均为加性秘密共享值，我们在上述百万富翁协议的基础上添加一些改进以构造安全比较协议。具体算法如下：

算法 3.1 SC $\rightarrow (\langle\delta\rangle_0, \langle\delta\rangle_1)$

输入: C_0, C_1 输入 $\langle x\rangle_i, \langle y\rangle_i, i \in [0,1]$.

输出: C_0, C_1 获得 $\langle\delta\rangle_0, \langle\delta\rangle_1$

1: C_0 生成随机数 $a, a \in \mathbb{Z}_N$, $\langle r\rangle_0 = \langle x\rangle_0 - \langle y\rangle_0 + a$，将 $\langle r\rangle_0$ 发送给 C_1

2: C_1 计算 $\langle r\rangle_1 = \langle x\rangle_1 - \langle y\rangle_1$，还原 r 值 $r = \langle r\rangle_0 + \langle r\rangle_1 = x - y + a$

3: C_0 和 C_1 使用百万富翁协议来比较 a 和 r，获得结果 $\langle v\rangle_i^B, i \in [0,1]$，$\langle v\rangle_i^B$ 代表 $\langle LT(a,r)\rangle_i^B$

4: C_i 选择随机数 $t_i \in \{0,1\}$，计算 $\langle v'\rangle_{1-i}^B = \langle v\rangle_i^B \otimes t_i$，$C_i$ 将 $\langle v'\rangle_{1-i}^B$ 发送给 C_{1-i}，因此 C_0 和 C_1 获取 $\langle v\rangle_i^B$ 的秘密共享值

5: C_0 和 C_1 计算 $\langle\mu\rangle_i^b \leftarrow \mathrm{MUL}(\langle v\rangle_1^B, \langle v\rangle_0^B)$

6: 两方计算 $\langle\delta\rangle_i = \langle v\rangle_i^B - 2\langle\mu\rangle_i^B, i \in \{0,1\}$

百万富翁协议无法直接比较秘密共享值，因此我们令云服务器 C_0 产生随机数 a，将秘密共享值 $\langle x\rangle$ 与 $\langle y\rangle$ 之间的比较转换为明文 a 与 $x - y + a$ 之间的比较。上

述转换只需要一轮交互，即可完成。

针对协议的输出，我们需要将布尔秘密共享值（$v = \langle v\rangle_0^B \bigoplus \langle v\rangle_1^B$）转变为加性秘密共享值（$v = \langle v\rangle_0^A + \langle v\rangle_1^A$）。根据如下观察，$v^A = \langle v\rangle_0^B + \langle v\rangle_1^B - 2\langle v\rangle_0^B\langle v\rangle_1^B$ 即可将布尔共享值转变为加性共享值。为计算 $\langle v\rangle_0^B\langle v\rangle_1^B$，我们令云服务器 C_i 与 C_{1-i} 秘密共享 $\langle v\rangle_i^B$，然后进行乘法操作。

3.4.3 安全最值计算协议

安全最值计算协议可以找到一组加性秘密共享值中最值的位置，给定 n 个秘密共享的数据，经过计算后，输出仅包含 0，1 秘密共享值的数组，其中最值位置存放 1，其他均为 0。若需要获得最值的具体数值，将结果数组与原始数组进行乘法运算后累加即可。

在 n 个数据中找到最值的传统方式为通过冒泡排序逐个比较大小，从而获得结果。上述方法需要进行 $n-1$ 次比较，并且不能并行计算，效率较低。本文提出的安全比较协议能够并行比较多对数据，基于此，我们针对数据集大小的场景分别设计了两种最值计算协议。

安全最值比较可以分为求解最大值和最小值，方法类似，这里我们以安全求解最小值为例进行阐述，求解最大值只需将协议中求解 $LT(x,y)$ 改为 $LT(y,x)$ 即可。

在处理小数据集时，我们可以通过增加并行比较的数据对，来减少比较的次数。以一个包含三个整数的数据集为例 $x_i \in \mathbb{Z}_N, i \in [1,3]$，我们令每个 x_i 与其他两个不同的数据比较。现在我们有 6 个比较数据对 $(x_i, x_{j\neq i}), i,j \in [1,3]$ 进行一轮数据比较。具体构造流程如下图所示：

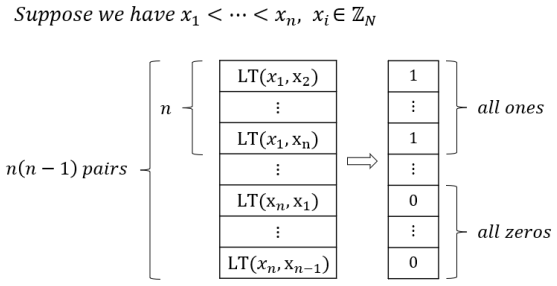

图 3.2　Secure minimal on small dataset (SMin(S))

结果标识为 $\{\langle\delta_1\rangle, ..., \langle\delta_6\rangle\}$，由云服务器 C_0 和 C_1 秘密共享。然后我们利用 MUL 来累乘与 x_i 相关的比较结果，获得 $\{\langle m_1\rangle, \langle m_2\rangle, \langle m_3\rangle\}$。假设 $x_1 < x_2 < x_3$，与 x_1 相关的比较结果全部为 1，同时与 x_2，x_3 相关的比较结果至少包含一个 0。因此，对应的累乘结果为 $m_1 = 1, m_2 = 0, m_3 = 0$。

针对大型数据集，上述方法增加的比较数据对以平方级剧增，带来大量冗余的通信和计算开销。因此，我们放弃增加比较数据对的思路，采取树型结构来减少比较轮次，在冒泡排序需要 $n-1$ 轮比较的基础上，减少为 $\log n$ 轮比较。

假设我们的数据集包含 n 个数据，我们首先比较 $n/2$ 对数据，然后用比较结果与原始数据相乘获取较小值。以 x，y 为例，比较结果为 $\delta = LT(x,y)$，较小值的计算方式为 $r = MUL(\delta, y) + MUL(1-\delta, x)$ 。经过上述操作数据集的大小由 n 变为 $n/2$，反复进行上述操作，直到集合仅包含一个数据，即最小值。比较过程如图3.3所示。

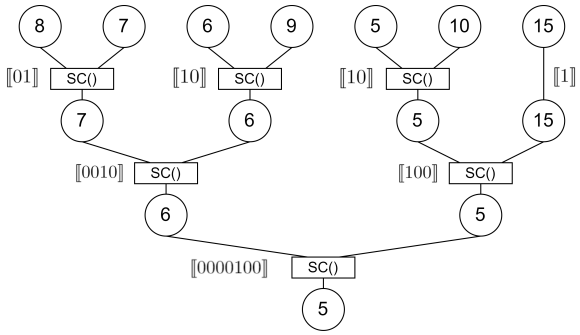

图 3.3　Secure minimal on large dataset (SMin(L))

3.4.4　安全过滤协议

本节我们将介绍如何在 kd-tree 数据结构上进行聚类，该方案的正确性验证由 ref-kd-tree 给出。自根节点开始，我们依次遍历所有节点来判断当前节点中包含的所有数据点是否可以被划分到某一个簇中。针对每个节点，我们维护一个候选簇集合，该集合中所有簇都可能是节点最终所属簇。在迭代中，我们不断删除不符合条件的候选簇知道仅剩一个候选簇，即节点所属簇。

算法如3.2所示。2-4 行，我们计算 kd-tree 每个节点到每个候选簇中心的距离，然后借助安全最值协议找到距离最近的候选簇标记为 z^*。7-12 行，我们执行一系列子协议来排除不符合候选条件的簇，即与候选簇 $z_j, z_j \neq z^*$ 相比，节点中包含所有数据点在空间上都离 z^* 更近，则我们认为 z_j 非候选簇。更加详细的规则解释在 ref-kd-tree 中给出。13-16 行，我们执行安全比较协议来判断候选簇集合是否仅剩一个簇。如果是，则将节点中所有数据划分到该候选簇中，并停止向下迭代，否则继续对子节点进行上述操作（17-18 行）。若聚类过程即将收敛，那么子节点可以不用进行冗余的距离计算和比较，节省大量计算资源。然而上述方案泄露了一比特数据来判断是否终止子节点迭代，这样少量的数据仅仅轻微泄露了迭代过程的信息，但是带来了效率的巨大提升。ref-p1-35-36 中采取了类似的方法作为一个交换来获取性能提升。

算法 3.2 SC $\rightarrow (\langle\delta\rangle_0, \langle\delta\rangle_1)$

输入: kd-tree $\langle tr\rangle$ 以及簇 $\langle z\rangle_j, j \in [k]$
输出: 新的簇中心 $\langle z'\rangle_j, j \in [k]$

for $i = 1$ to n do
    for $j = 1$ to k do
        $\langle d\rangle_j \leftarrow \text{SED}(\langle z\rangle_j, \langle c\rangle_i)$
    end for
    $\{\langle r\rangle_1, ..., \langle r\rangle_k\} \leftarrow \text{SMin}(\langle d\rangle_1, ..., \langle d\rangle_k)$
    计算最近候选簇 $\langle z^*\rangle \leftarrow \sum_{p=1}^{k} \text{MUL}(\langle r\rangle, \langle z\rangle_p)$
    for $j = 1$ to k do
        $\langle r\rangle \leftarrow \text{SC}(\langle u\rangle, 0)$, $\langle \mu\rangle \leftarrow \langle z^*\rangle - \langle z\rangle_j$
        $\langle v\rangle \leftarrow \text{MUL}(\langle r\rangle, \langle c\rangle_{min}) + \text{MUL}(1 - \langle r\rangle, \langle c\rangle_{max})$
        $\langle d^*\rangle \leftarrow \text{SED}(\langle z^*, \langle v\rangle)$, $\langle d_j\rangle \leftarrow \text{SED}(\langle z\rangle_j, \langle v\rangle)$
        $\langle r\rangle_j \leftarrow \text{SC}(\langle d^*\rangle, \langle d\rangle_j)$
    end for
    $\langle f\rangle \leftarrow \text{SC}(\sum_{j=1}^{k} \langle r\rangle_j, 1)$
    if $\langle f\rangle_0 + \langle f\rangle_1 == 1$ then
        $\langle \mu\rangle_j \leftarrow \langle \mu\rangle_j + \text{MUL}(\langle r\rangle_j, \langle c\rangle), j \in [k]$
    else
        $\langle z\rangle_j \leftarrow \text{MUL}(\langle z\rangle_j, \langle r\rangle_j), j \in [k]$
        pass candidate cluster set $\{\langle z\rangle_1, ..., \langle z\rangle_k\}$ to child notes and repeat above process
    end if
end for

3.5 基于 kd-tree 的隐私保护 k-means 方案

本节主要介绍基于 kd-tree 的隐私保护 k-means 方案（PPOKC）的具体细节。我们假设数据在双云服务器 C_0 和 C_1 上加性秘密共享，并且二者不可以合谋。云服务器在密文基础上进行系列安全协议获取聚类结果，整个过程可以被划分为两个阶段：

- **初始化**: 首先，用户基于拥有的明文数据构建 kd-tree。然后将数据秘密共享为两份，并分别发送给云服务器 C_0 和 C_1。此外，原始数据也会在被划分后发送给云服务器以支持其他计算。此后用户不再参与聚类过程。
- **聚类**: 对 kd-tree 中节点执行过滤操作，根节点的候选簇集合包含所有初始簇。针对树中的节点，我们执行过滤操作，直到所有数据都被划分到对应的簇。在每轮迭代结束，我们更新簇中心并且将新簇与旧簇相比较来判断是否满足

聚类收敛条件。

3.5.1 算法详述

PPOKC 算法的细节在3.3中给出，虽然我们这里提供了一种判断是否停止迭代的方法，在稍后的实验中我们会采取固定迭代次数的方式来测试性能。不同的数据集迭代收敛所需的轮次通常不同，目前大量相关隐私保护聚类研究采取的迭代终止策略通常存在安全问题，即泄露数据集迭代收敛所需轮次。同时，k-means 聚类算法在迭代足够多轮次后一定会收敛 ref-p1-4。

算法 3.3 隐私保护外包聚类算法

输入: 明文数据 $x_{ij}, i \in [n], j \in [m]$

输出: 秘密共享的簇中心 $\langle z'_j \rangle, j \in [k]$

1: **初始化（用户）**
2: 在明文基础上构造 kd-tree
3: 加性秘密共享数据 x_{ij} 以及 kd-tree，随机选择初始簇中心 $z_j, j \in [k]$，上述内容分别发送给 C_0 和 C_1
4: **聚类（C_0, C_1）**
5: $\{\langle z'_1 \rangle, ..., \langle z'_k \rangle\} \leftarrow \text{SF}(\langle z_1 \rangle, ..., \langle z_k \rangle)$
6: $\langle r \rangle \leftarrow \sum_{i=1}^{n} \sum_{j=1}^{m} \text{SC}(\langle z_{ij} \rangle - \langle z'_{ij} \rangle, 1)$
7: if $\langle r \rangle_0 + \langle r \rangle_1 == 1$ then
8: 终止迭代，返回结果给用户
9: else
10: $\{\langle z_1 \rangle, ..., \langle z_k \rangle\} \leftarrow \{\langle z'_1 \rangle, ..., \langle z'_k \rangle\}$
11: 回到步骤 5
12: end if

3.5.2 讨论

最近，[15] 提出了一种双云环境下基于密文打包技术的高效隐私保护 k-means 聚类方案。然而，我们发现虽然方案非常高效，但是存在一些安全性问题。在论文[15] 的 S3ED 算法中 11 行，我们可以看到 $Dst_{aj} \leftarrow Dst_{aj} * r_1^a + r_1^a$, for $j \in [k]$，意味着距离矩阵中行 a 中所有值均被添加相同的噪声。根据[17] 章节 7.3 的分析，我们发现上述方式无法保护数据的分布信息。在本方案中，不同的距离均为秘密共享值，在进行安全比较时，添加了互异的随机噪声，因此不会泄露原始数据任何相关信息。

3.6 安全性分析

本节中，根据[18]中的标准模拟理论，我们假设双云服务器 C_0 和 C_1 为潜在的攻击者，并提供对于安全欧式距离，安全比较协议、安全最值协议、以及安全过滤协议的书面安全性证明。然后，根据组合理论，我们证明 PPOKC 方案在半诚实模型下是安全的。为了论证方案的安全性，我们首先介绍半诚实模型下的安全性定义[19]：

定义 3.1: *参与方 A 和 B 在半诚实模型下计算函数 f，若存在一对非均匀的概率多项式模拟器 S_A, S_B，以使得对于每一个安全参数 n 和所有输入 $x, y \in \{0,1\}^n$，都有以下内容成立：*

$$\begin{aligned}\{\mathcal{S}_A(x, f(x,y)), f(x,y)\} &\approx \{e \leftarrow [A(x) \leftrightarrow B(y)] : \mathrm{View}_A(e), \mathrm{Out}_B(e)\} \\ \{\mathcal{S}_B(y, f(x,y)), f(x,y)\} &\approx \{e \leftarrow [A(x) \leftrightarrow B(y)] : \mathrm{View}_B(e), \mathrm{Out}_A(e)\}\end{aligned} \tag{3.6}$$

此外，我们需要用到如下引理：

引理 3.1: *如果一个随机元素 c 在 $\mathbb{Z}_N$ 上是随机分布的，且与 $\mathbb{Z}_N$ 中的元素 x 无关，则 $c \pm x$ 在 $\mathbb{Z}_N$ 也是均匀随机分布的，与 x 的值无关。*

3.6.1 安全计算模块安全性分析

定理 3.1: *安全欧式距离计算协议在半诚实模型下是安全的。*

证明：*在我们的安全欧式距离计算协议中，大部分计算由云服务器 C_0 和 C_1 分别在本地进行，没有泄露可能性。交互操作仅在乘法部分，该方法的安全性证明已经由[12]给出。因此，根据组合理论，本协议的安全性得证。*

定理 3.2: *安全比较协议在半诚实模型下是安全的*

证明：*该算法是基于[20]中的百万富翁协议，对输入和输出进行改造。C_0 用不同的随机数打乱数据，然后发送给 C_1，这种方式根据引理3.1是安全的。输出部分，由布尔秘密共享值转换为加性秘密共享值的安全证明依赖于乘法的安全性。因此，我们证明安全比较协议是安全的。*

定理 3.3: *安全最值协议在半诚实模型下是安全的。*

证明：*小数据集上的安全最值协议是安全比较协议的变形，*

3.7 系统评估与性能分析

我们前述协议和方案的基础上实现了高效的隐私保护 k-means 系统，并在本节总结分析该系统理论与实验性能。我们在多个人工合成的数据集上进行实验以观察系统的实用性和可扩展性。同时，我们也在现实数据集进行了充分的实验，以和目前最先进的隐私保护 k-means 聚类方案[15][21] 进行实验对比。

3.7.1 理论分析

本节中，我们主要从理论层面分析方案的计算和通信开销。安全欧式距离仅包含简单的秘密共享加法和乘法运算，因此不纳入分析。安全比较算法主要基于[20] 中的百万富翁协议，详细的效率分析已经在论文中给出，这里不再赘述。因此我们主要分析安全最值协议以及安全过滤协议。值得注意的是，乘法所需三元组的计算是离线进行的。

3.7.1.1 计算开销

考虑到上述两个协议，主要由安全比较和秘密共享乘法加法组成，乘法与加法相对安全比较协议来说计算开销可以忽略不计，因此我们在以下的论述中，以安全比较协议的运行次数为基准给出分析结果，如表??所示：

表 3.1 Computation Overhead

Protocol	SMin(L)	SMin(S)	SF
SC	$\log n$	1	$2kn$

借助安全比较协议，我们可以运行一轮协议比较多对数据，效率非常高。然而，随着比较数据的增加，计算开销也随之剧增。虽然小数据集上的安全最值协议只需要运行一次安全比较协议，但是当数据量变得很大时，协议运行非常耗时。因此我们设计了另一种能够减少比较数据量略微增加比较次数的协议。

3.7.1.2 通信开销

假设我们的数据为 $l-$ 比特，我们仅分析乘法的通信开销，因为安全比较协议的通信开销分析比较复杂，并且在[20] 中已经给出。在安全比较协议，以及小数据集上的安全最值协议，通信开销分别为 $3nl$，$7n(n-1)l$ 比特。对于大数据集上的安全最值协议，开销最多不超过 $nl\log n$ 比特。因为安全过滤协议的开销分析非常复杂，我们这里仅给出近似值，$O(n^2\log n)l$ 比特。

3.7.2 实验分析

系统由 c++ 实现，选择两种不同的机器进行实验。

为了解方案的可拓展性，我们在一台服务器模拟双云对人工数据集进行实验。该机器的配置为 Intel Core i9-10980XE 3.00GHz CPU and 64GB RAM。

为了直观的与[15] 的实验数据进行比较，我们找到论文中所提到的机器进行对比实验。该机器的配置为 Inter Core i7-7700HQ 2.80 GHz CPU and 16 GB RAM。

3.7.2.1 数据集

为了能够进行全面的对比，我们使用了表格3.2所示数据集，其中每一个都在之前的研究工作中出现过：

- KEGG 是一个集成的数据库，主要包含 15 个不同的数据库资源，包含的内容主要是分子系统的计算机表示，从基因信息到化学信息等。本方案选择的 KEGG Metabolic Reaction Network（Undirected），是收录于 UCI KDD 条目的现实世界数据集[22]。我们从中提取 8192 条数据，5 个维度（$n = 8192, m = 5$），令簇个数 $k = 3$ 进行实验，以和论文[15] 保持一致。
- Lsun 数据集是来自论文[23] 的一个人造二维数据集，该数据集常用于聚类研究以进行对比分析和展示。其中包含 400 个数据，3 个簇。我们将用于与论文[24] 的比较。
- 我们构造了数据量为 n，维度为 m 的人造数据集，随机值为整数，均匀分布在 $[0, 1000]$ 内。簇的个数在 3 到 15 之间以方便进行实验。

表 3.2 数据集详细信息

数据集名称	数据集大小 n	簇个数 k	维度 d
Lsun	400	3	2
KEGG	8192	3	5
Synthetic	$\{10000, 60000\}$	$\{3, 15\}$	$\{2, 20\}$

3.7.2.2 现实数据集上的实验

与论文[15] 方案对比：为了进行最直接的比较，我们使用相同的数据集和一致的实验机器运行本系统。实验数据如表格3.3所示，其中给出了 PPCOM[25]、SEOKC[15] 以及本方案 PPOKC 进行一轮迭代的通信和计算开销。

从计算耗时角度来看，PPCOM 方案和 SEOKC 方案迭代一轮分别需要 401 分钟和 17s，而我们的 PPOKC 方案则需要 20s，对比来看，比 PPCOM 方案要快 1203 倍，比 SEOKC 方案慢 3s。从通信开销角度来看，PPCOM 需要 1430MB 而 SEOKC

表 3.3　scheme comparison ($n = 8192, m = 5, k = 3$)

Performance	PPCOM	SEOKC	PPOKC
计算开销	401 min	17 s	20 s
通信开销	1430 MB	148 MB	405MB

需要 148MB，本方案则需 405MB。

综合来看，我们的方案的表现远远超过 PPCOM，接近目前最高效的 SEOKC 方案。值得一提的是，从3.5.2我们可以看到，SEOKC 方案虽然高效但是存在不容忽视的安全性问题，会暴露数据分布。而 PPOKC 则更加安全，不会泄露相关信息，因此性能上略微逊色是可以接受的。

与论文[24]的比较：为了进行公平的比较，我们在 Lsun 数据集上进行实验。对比论文的机器为 Inter i7-3770 3.4GHz 20GB RAM，我们的前述实验机器要慢 1.3 倍。

论文[24]提供了 3 种不同的隐私保护 k-means 聚类方案，分别是精确的、稳定的以及近似的方案。上述方案进行 15 轮迭代分别用时为 545.91 天、48.9 小时、15.47 小时。在我们的系统中，完成相同的迭代轮次总共需要 20.94 秒，用时远远少于前述三种方案。

此外，我们设置相同的初始簇中心初始化，在 Lsun 数据集上分别运行明文 k-means 方案以及本文提出的 PPOKC 方案，分类结果如图3.4和3.5所示。本方案所有安全协议的正确性均以论文[7]中提出的基于 kd-tree 的改进 k-means 方案为支撑。因此，最终聚类结果和明文方案完全一致。

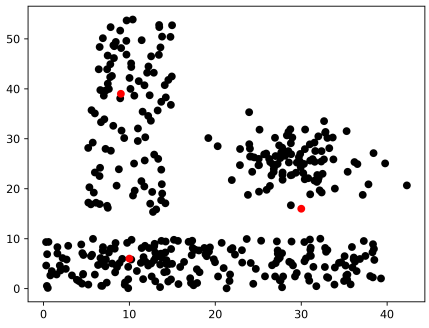

图 3.4　Plaintext K-means model

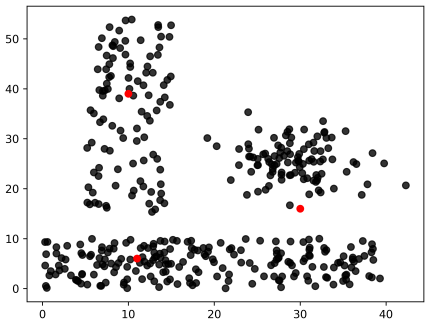

图 3.5　Privacy Preserving model

3.7.2.3　人造数据集上的实验

本节，我们在数据量跨度为 $n \in [10000, 60000]$ 的人造数据集上进行系列对比实验。我们在生成数据时控制簇的个数在 3-15 之间，彼此之间没有重叠的部分。

同时，聚类终止条件选择达到指定迭代次数即停止聚类比较结果。该选择主要原因如下：不同数据集收敛所需迭代次数不固定，受初始簇中心的影响较大，若按照算法中所给出的判断是否收敛来停止迭代，会导致时间分析不准确。

我们考虑三个影响聚类的参数，分别是数据集大小 n、数据的维度 m 以及簇的个数 k。为了准确的分析参数对于方案的性能影响，我们的策略是固定一个参数，然后变化另外两个参数的值来运行系统。实验运行的计算和通信开销如图3.6和3.7所示。

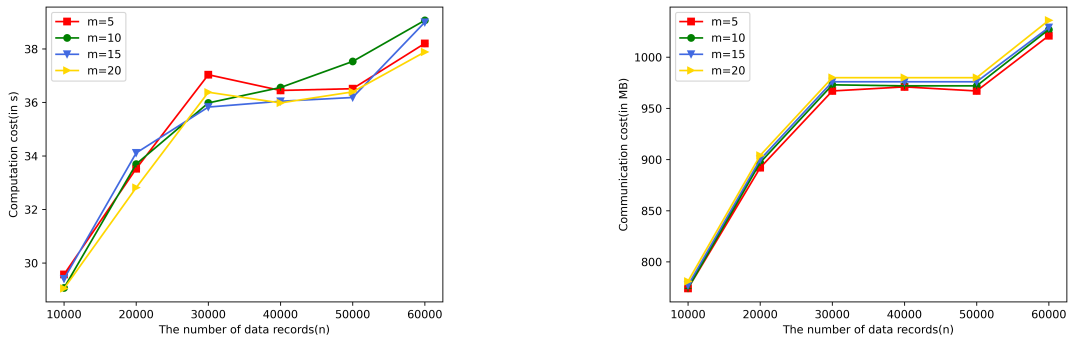

图 3.6 $k=3, T=20$

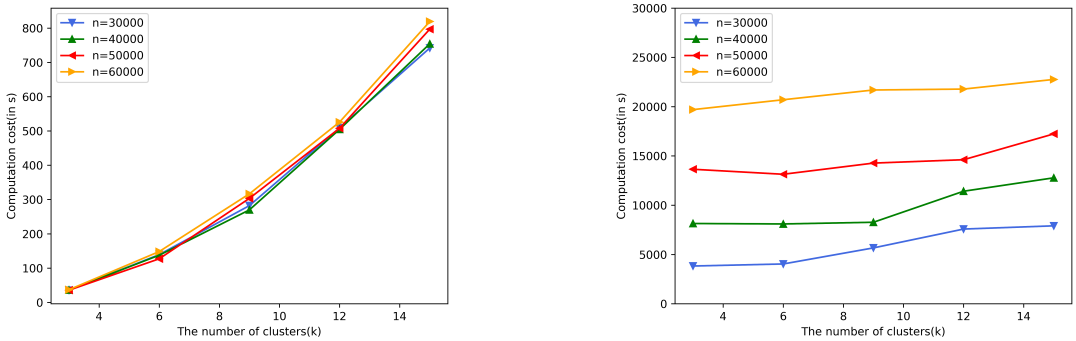

图 3.7 $m=5, T=20$

对于图3.6，我们可以看到计算和通信开销随着数据量 n 增加而增加。然而，数据维度 m 的变化对于计算和通信开销的影响则非常小，因为维度多少只会影响计算中的乘法和加法次数，而不会影响比较的次数，后者才是方案的瓶颈。乘法操作可以通过并行进一步优化。

对于图3.7，计算开销随着 k 的变化呈线性变化，而 k 对通信开销的影响则较小。值得一提的是，给定相同的 k 值，不同的数据量对于计算开销的影响较小。

通过分析实验结果，我们发现本文提出的方案兼具高效与安全性，并且能够适应大型数据集。与研究[15][25] 相比，我们的方案在 k 较大时，实验表现更好。基

于 kd-tree 的改进 k-means 算法允许我们减少大量冗余计算，进一步加速聚类过程，此外，SIMD 操作可以很轻松被引入方案来提升效率。

3.8 本章小结

隐私保护 k-means 聚类方案的研究通常难以兼具效率与安全性，实现完全隐私保护的方案耗时通常难以接受[24]，高效的方案则可能存在泄露隐私的风险[15]。过去研究常用到的密码学工具为秘密共享[21] 和同态加密[24][15]，二者各有优缺点。秘密共享技术进行加法和乘法运算较快，但是通信开销巨大，而同态加密则计算开销较大，通信开销较小。

为了能够设计兼具安全与高效的隐私保护 k-means 聚类方案，我们在秘密共享的基础上，提出了一个基于 kd-tree 的隐私保护外包 k-means 聚类方案。方案利用了 kd-tree 数据结构来减少冗余计算，提升聚类效率，同时保证聚类的正确性。此外，我们基于秘密共享设计了一系列隐私保护的计算协议，这些协议独立于整个方案，可以灵活的用于其他各种基于秘密共享的隐私保护方案中。实验结果证明，我们的方案在保证聚类结果正确的基础上，保护了数据的隐私安全，减少了计算和通信开销，为用户提供了一种实际可行的外包聚类方案。

然而，聚类通常是对海量数据进行分析挖掘，数据量级可以达到亿万级别。在该场景下，即便是目前最高效的隐私保护聚类方案所需时间也是不可接受的。因此需要探索新的方案设计方向，即近似的隐私保护方案。机器学习对于计算的精度要求通常较宽松，因此我们可以考虑通过设计近似安全计算协议来简化计算过程，提升效率。

第四章　隐私保护密度聚类方法研究

4.1　引言

海量可获取的数据以及云计算提供的计算能力驱动机器学习的快速发展。有监督学习例如神经网络，使用带标签的数据来训练具有识别能力的模型。与之相反的是无监督学习，没有训练模型的过程，旨在从未标记的数据中发现未知的模式和特征。聚类是一种广泛使用的无监督机器学习工具，能够将相似的数据划分到同一个集合中。实际应用中，聚类常用于许多数据安全极为重要的领域，例如商业数据分析，市场数据挖掘以及医院病理信息分析等。上述场景中，数据一旦泄露会造成极大经济损失。

目前，已有许多关于隐私保护聚类方案的研究，其中数量最多的是与 k-means 相关的研究 ref-sok。然而 k-means 聚类过程相对比较简单，并且只能检测到凸型簇，因此适用数据集类型极为受限。此外，簇的数量 k 必须要根据专业领域知识提前给出，如果只了解数据集的子集难以确定 k 的值。同时，k-means 聚类不包含噪声的概念，并且聚类的结果对异常值非常敏感因为每一个输入都要被划分到某一个簇中。因此，即便某个输入不属于任何一个簇，它都会被划分到距离最近的簇，影响该簇的中心位置（簇中所有数据的平均值）。

为了解决上述隐私保护方案中存在的问题，这里我们引入对隐私保护 density-based spatial clustering of applications with noise（dbscan）的研究。dbscan 是一种由 ref-dbscan-ester20 提出的更加灵活的聚类算法，能够检测到任意形状的簇。此外，簇的数量是根据数据集的特点灵活产生的，无需人工确定。该算法对异常值不敏感，会将其标记为噪声。本章首先提出了方案一，一种安全高效的隐私保护 dbscan 方案，该方案针对明文算法进行改进以适应安全计算的特点，将复杂度降低为 $O(n^2)$，显著降低聚类所需时间。

传统 dbscan 方案存在数据划分结果不稳定的问题，即最终划分结果取决于算法运行中数据遍历的顺序，将数据打乱后重新进行聚类，同一个点可能会被划分到不同的簇中。为了解决该问题，在 ref-redbscan 论文思想的基础上，我们在方案二中提出了改进的隐私保护 dbscan，以获取稳定的聚类结果。

尽管 dbscan 具有诸多优点，其聚类过程涉及两个重要参数 MinPts 和 Eps，这两个参数的取值与数据分布密切相关，通常需要人工分析后给出。为解决该问题，在论文 ref-dbhc 的基础上，我们在方案三中提出了基于 dbscan 的隐私保护层次聚类，借助 k 近邻算法和 k 线图来决定聚类参数，并进行多轮聚类来适应不同密度的簇。本章的组织结构如下：第4.2节介绍了 dbscan 的相关内容。第4.3节中描述了

系统模型、安全模型以及设计目标。第4.4节中增加了一些基于秘密共享的隐私保护计算模块。第4.5节对隐私保护 dbscan 方案进行了详细介绍。第4.6节中提出了方案二改进的隐私保护 dbscan 方案。第4.7节中阐述了基于 dbscan 的隐私保护层次聚类方案。第4.8节中从理论上分析了方案的正确性和安全性。第4.9节中对方案进行了实验评估。最后，在4.10节对本章进行了总结。

4.2 预备知识

本节我们主要介绍 DBSCAN 的明文算法以及相关的改进算法。

4.2.1 DBSCAN

Density-based Spatial Clustering of Applications with Noise(DBSCAN) 是一种基于密度的聚类算法，在密集区域聚集在一起的数据点被划分到同一个簇中，稀疏区域的数据被标记为噪声[26]。算法要求两个重要参数：

- ϵ：确定两个点被视为相邻的最大距离
- MinPts: 确定领域内至少包含多少点才能构成一个簇

DBSCAN 围绕每个数据点以 ϵ 为半径构建圈，并将其划分为核心点、边界点以及噪声点。若圈内不包含任何点，则认为是噪声点。若数据点圈内包含点的数量至少为 MinPts 个，则认为是核心点，否则为边界点。围绕上述定义展开，DBSCAN 还包含如下概念：

假设样本集为 $D = (x_1, x_2, ..., x_m)$：

- 密度直达：若 x_i 位于 x_j 的 $\epsilon-$ 邻域内，且 x_j 为核心对象，则称 x_i 由 x_j 密度直达，反之不一定成立。
- 密度可达：对于 x_i 和 x_j，若存在样本序列 $p_1, p2, ..., p_t$，满足 $p_1 = x_i, p_T = x_j$，且 p_{t+1} 由 p_t 密度直达，则称 x_j 由 x_i 密度可达。
- 密度相连：对于 x_i 和 x_j，如果存在核心点 x_k，使得 x_i 和 x_j 均由 x_k 密度可达，则称 x_i 和 x_j 密度相连。

下面我们具体阐述 DBSCAN 聚类算法的流程：

1. 初始化核心点集合 $\Omega = \varnothing$，初始化簇数量 $k = 0$，初始化未访问点集合 $\Gamma = D$，簇划分 $C = \varnothing$
2. 遍历所有点 $i = 1, 2, ..., m$，按如下方式找到所有核心点：
   (a) 计算距离，找到样本 x_i 的 $\epsilon-$ 邻域内所有点集合 $N_\epsilon(x_i)$
   (b) 如果集合包含数据点个数满足 $|N_\epsilon(x_i)| \geq MinPts$，将 x_i 加入核心点集合：$\Omega = \Omega \cup \{x_j\}$

3. 若 $\Omega = \varnothing$，则算法结束，否则转入步骤 4

4. 在 Ω 中，随机选择一核心点 o，初始化当前簇包含核心点集合 $\Omega_c = \{o\}$，簇序号为 $k = k + 1$，当前簇包含点集合 $C_k = \{o\}$，更新未访问样本集合 $\Gamma = \Gamma - \{o\}$

5. 若 $\Omega_{cur} = \varnothing$, 则簇 C_k 聚类完毕，更新簇集合 $C = \{C_1, C_2, ..., C_k\}$，更新核心点集合 $\Omega = \Omega - C_k$

6. 从 Ω_{cur} 中取出一个核心点 o'，通过邻域距离 ϵ 找到所有 $\epsilon-$ 邻域子集 $N_\epsilon(o')$，令 $\Delta = N_\epsilon(o') \cap \Gamma$，更新当前簇包含点集合 $C_k = C_k \cup \Delta$，更新未访问点集合 $\Gamma = \Gamma - \Delta$，更新 $\Omega_{cur} = \Omega_{cur} \cup (\Delta \cap \Omega) - o'$，转入步骤 5

7. 输出结果为：簇划分 $C = \{C_1, C_2, ..., C_k\}$

DBSCAN 能够应用于任意维度的数据集。铭文上最坏情况下的复杂度为 $O(n^2)$，其中 n 为数据的数量。

4.2.2 改进的 DBSCAN

正如提出 DBSCAN 的论文[27] 中所说，该算法在检测相邻簇的边界数据点时划分结果不稳定。若相邻簇 C_1 与 C_2 的边界上有一数据 x，按照划分规则，该点既可以被划分到 C_1，也可以被划分到 C_2，最终划分结果取决于算法运行时 x 被处理的顺序。

在[28] 中，作者认为边界对象对于传统 DBSCAN 算法簇的扩展没有贡献，因此可以改造算法的形式，以使得边界点的划分结果稳定。文章提出在聚类过程中，暂时不处理边界对象，直到所有核心对象都被分配到对应的簇中。最后，将所有未被划分的边界对象划分到最近的核心对象所属簇中即可。

4.2.3 基于 DBSCAN 的层次聚类

传统 DBSCAN 算法中，MinPts 和 ϵ 的取值对于聚类效果有极大影响，但同时又难以确定，因为二者受数据分布的影响较大。为了解决这个问题，论文[29] 提出一种基于 DBSCAN 的层次聚类方案。首先借助 k 近邻算法和 k 线图来决定两个参数。因为实际数据大多不是均匀分布的，需要计算不同的 ϵ 值。然后，根据不同的 ϵ 值来多次聚类。最后，若最终结果所得簇个数超过了用户实际认可的数量，则选择不同的簇进行合并，直到满足要求。该方案经过实验，取得了较好的效果。

4.3 问题描述

4.3.1 系统模型

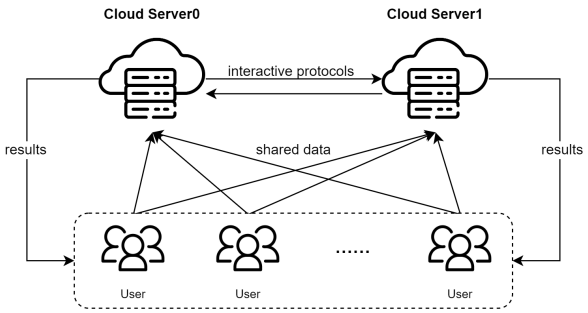

图 4.1 System Model

如图4.1所示，我们的系统由两个部分组成：服务器端和用户端。与图3.1类似，服务器端为两个云服务器，获取用户数据后执行一系列安全协议，最后分发结果。

不同的是，图3.1中客户端为某一个持有全部明文数据的组织机构，本方案系统模型中的用户既可以是单一机构，也可以是各自持有不共享数据的不同机构。第二种场景下，用户可以是不同的医疗机构，期望能够在不泄露患者信息的情况，联合起来对医疗诊断或者症状信息进行聚类来做数据挖掘与分析。用户将数据进行秘密共享后发送给双云服务器即可。在获取到聚类结果后，不同机构各自执行简单的算法来还原结果。

4.3.2 安全模型

本方案基于半诚实模型的假设，细节说明见3.3.2中的说明。具体来说：

对用户而言，不同用户可能会根据聚类结果，推断其他用户输入数据的具体内容。

对服务端而言，可能会试图根据秘密共享的数据、中间计算结果以及聚类结果推测原始内容。

4.3.3 设计目标

我们的目标是设计三种针对不同场景的隐私保护 DBSCAN 方案，具体而言，设计目标如下：

- 能够高效的在密文上运行朴素的 DBSCAN 算法，在效率略微降低的前提下，运行结果稳定的隐私保护 DBSCAN 算法。在运行效率可接受的情况下，运行参数无需提前设置的基于 DBSCAN 的层次聚类。
- 三个密文方案运行结果与对应明文方案一致，精度无损失。

- 云服务器上所有数据均在密文状态，并且无法被利用来还原初始数据信息。不同用户之间无法知悉彼此的数据和聚类结果。

4.3.4 系统输入

数据持有者首先利用加性秘密共享来将数据划分为两份，分别发送给两个云服务器。采用加性秘密共享分发数据的方式允许系统中加入任意多个数据持有者提供聚类数据。此外，我们的系统能够支持任何数据划分方式，例如水平划分、垂直划分以及混合划分。水平划分方式指的是，不同用户分别持有完整但是互异的数据[30]，垂直划分方式则是指不同用户持有相同数据记录的不同参数[31]，混合划分方式则是上述两种方式的结合[32]。

4.4 基于秘密共享的隐私保护计算模块

在3.4节所述算法的基础上，我们根据隐私保护 DBSCAN 方案的特点设计了如下计算模块。

4.4.1 安全排序协议

安全排序协议主要由安全比较协议构成，用户输入秘密共享的数组后，我们期望能够得到按照升序排列的密文结果。协议的核心思想为，将 n 个数据上的排序拆分为 n 次求解序列最值。

具体如算法4.1所示，我们进行 n 次迭代，2 行开始求解最小值，结果为秘密共享值 $\langle t\rangle_i \in \{0,1\}, i \in [1,n]$，数组中最小值的位置为 1，其余均为 0。4-6 行开始遍历每个结果，res 通过累加记录最小值的具体值。6 行的目的是更新原始数组，通过将最小值变为全局最大值，以消除对后续计算的影响。若当前元素为最小值（即 $\langle t\rangle_i = 1$ ），则对应 $D[i]$ 中的值变为最大值 maxv，若不是则保留原值，不做修改。

4.5 隐私保护 dbscan 方案

本节提出的隐私保护 DBSCAN 方案主要分为三个阶段，具体介绍如下：

- 初始化：通过获取的密文数据，初始化存储在云服务器上的自定义数据结构体（以下简称元素），然后计算不同元素之间的距离关系，并判断元素是否为核心点。
- 聚类：第一阶段获取的信息，我们提出了一种全新的聚类方法，引入临时簇这一概念，通过记录临时簇是否相连的信息来聚类。
- 还原结果：用户获取聚类记录的临时簇相连信息，通过深度优先遍历算法来还原数据簇划分结果。

算法 4.1 安全排序协议

输入: $D = \{\langle x\rangle_1, \langle x\rangle_2, ..., \langle x\rangle_n\}$

输出: sorted result$D'\{\langle x'\rangle_1, \langle x'\rangle_2, ..., \langle x'\rangle_n\}$

```
1: for i = 1 to n do
2:         {⟨t⟩_1, ...⟨t⟩_n} ← smin(D)
3:         res ← 0
4:         for j = 1 to n do
5:                 res ← res + mul(⟨t⟩_j, D[j])
6:                 D[j] ← mul(D[j], 1 − ⟨t⟩_j) + mul(maxv, ⟨t⟩_j)
7:         end for
8:         r[i] ← res
9: end for
```

4.5.1 初始化

对于每个秘密共享的数据，我们构造 sharedElem 对象。$cluId, cluId \in [1, n]$ 为初始簇中心，值取该数据在数组中的下标，比如第一个数据初始化 clusterId 为 1。$isMark, isMark \in [0, 1]$ 表示该数据对象是否已经被划分到簇中。data 则存储对应的秘密共享值，为一个向量。最后 $isCore, isCore \in [0, 1]$ 标识数据对象是否为核心点。

```
1: classsharedElem(sdata,idx):
2:         cluId = idx
3:         isMark = 0
4:         data = sdata
5:         isCore = 0
```

构造完 sharedElem 对象后，我们得到了 sharedList 数组。在初始化阶段，我们计算对象之间的距离，获取相邻关系，同时判断是否为核心点。具体过程如算法4.2所示。1 行，我们构造数组 cnt 存储每个元素邻域范围内包含其他元素的个数。2-7 行，我们计算距离信息，将第 i 个元素与其他所有元素的距离 $d_j, j \in [n]$ 与 eps 相比较，判断二者是否相邻, 结果存储到二维数组 $u_{ij}, i, j \in [n]$ 中，最后统计元素 i 邻域范围内包含点的个数。9 行，我们并行比较邻域点个数 cnt 与 $minpts$ 的大小关系，并更新 sharedList 的 $isCore$ 属性。

4.5.2 聚类

初始化之后，我们获得了元素的 $isCore$ 信息，以及元素之间的相邻关系 $u_{ij}, i, j \in [n], u_{ij} \in \{0, 1\}$。接下来，我们将具体介绍一种基于 DBSCAN 的全新聚

算法 4.2 初始化

输入: 数组 sharedList
输出: 相邻关系二维数组 u，更新后的 sharedList

```
1: cnt ← [0, ..., 0]_n
2: for i = 1 to n do
3:         for j = 1 to n do
4:                 d_j ← SED(sharedList[i], sharedList[j])
5:         end for
6:         u_i ← sc(d, [eps, ..., eps]_n)
7:         cnt ← sum(u_ij) for j ∈ [n]
8: end for
9: r ← sc(cnt, [minpts]_n)
10: for i = 1 to n do
11:         sharedList[i].isCore = r[i]
12: end for
```

类方式。该算法的核心在于构造若干临时簇，通过记录临时簇之间的连接关系，最后由用户来还原最终簇划分结果。

在初始化的过程中，我们默认每个元素的初始 $cluId$ 为下标，即每个元素都默认为独立的临时簇。在聚类的过程中，对于核心点，我们修改其 ϵ 范围内所有未标记点的 $cluId$，并将状态修改为已标记。对于所有非核心点，无法修改其他点的属性，并且被标记后，无法再被其他核心点标记。经过上述过程，数据集已经被划分为若干临时簇。根据 DBSCAN 的原理可知，单个簇由若干密度可达的核心点展开。通过记录核心点之间的相邻关系，我们在还原结果的阶段，将相连的临时簇合并，即可得到最终的结果。

具体流程如4.3所示，为了简化说明，我们将输入数组简写为 d，并将结果记录在 plist 数组中，数组中每个元素的结构依次为，元素 i 的 $cluId$，元素 j 的 $cluId$，临时簇 1 和簇 2 是否相连。2 行，更新元素 i 的 isMark 标识，若元素 i 是核心点（$isCore = 1$）或者已经被划分（$isMark = 1$），则令 $d[i].isMark$ 的值为 1，这里的计算逻辑体现的是逻辑或关系。3-8 行，我们开始对其他所有元素进行处理。4-6 行的计算逻辑是，若元素 i 与元素 j 相邻（$u_ij = 1$），且元素 j 未被标记过（$d[j].ismark = 0$），同时元素 i 为核心点（$d[i].isCore = 1$），则元素 j 的簇标识修改为与元素 i 的簇标识一致，否则不做修改。7 行，记录临时簇相连信息，若 $d[i]$ 和 $d[j]$ 同为核心点，并且相邻 $u_{ij} = 1$ 则二者相连。8 行，我们更新 $plist$ 的信息，记录对应值。

算法 4.3 聚类

输入: 数组 sharedList 简写为 d，相邻关系 $u_{ij}, i, j \in [n]$
输出: 簇连接关系 $plist$

1: for $i = 1$ to n do
2: $d[i].isMark \leftarrow d[i].isCore + d[i].isMark - mul(d[i].isCore, d[i].isMark)$
3: for $j = 1$ to n do
4: $m_1 \leftarrow mul(d[i].isCore, u_{ij})$
5: $m_2 \leftarrow mul(1 - d[j].isMark, m_1)$
6: $d[j].cluId \leftarrow d[j].cluId + mul(m_2, d[i].cluId - d[j].cluId)$
7: $d[j].isMark \leftarrow d[j].isMark + m_1 - mul(d[j].isMark, m_1)$
8: record new element in plist $(d[i].cluId, d[j].cluId, mul(m_1, d[j].isCore))$
9: end for
10: end for

4.5.3 还原聚类结果

聚类后，我们在 plist 中记录了所有临时簇的相邻关系。云服务器分别将自己持有的份额分别发送给用户还原结果。在4.5中，我们详细说明如何还原结果。1-5 行，我们构造大小为 $n \times n$ 的全零二维数组，来存储邻接关系，若相邻，则对应值大于零。6 行，我们初始化一个数组来标记每个数据是否已经被划分，若已经被划分，则不再继续处理。7 行，我们构建最终结果的簇编号，自 1 开始累加。8 行开始遍历，若元素 i 未被划分，则更新 mark，开始构建新簇，执行深度优先遍历找到所有与 i 相连的元素纳入到新簇中。dfs 的过程如4.4所示，边递归边记录映射关系。最后，根据映射关系，我们更新所有元素的最终簇编号。

4.6 改进的隐私保护 dbscan

本节我们将在上一节的基础上详细阐述一种改进的隐私保护 DBSCAN 方案，该方案能够在数据集被打乱的情况，使满足条件可以被划分到多个簇的数据点稳定分类到最近的簇中。

方案包含三个步骤，初始化、聚类以及还原结果。还原结果的过程与4.5.3一致，这里不多做说明。主要区别在于初始化和聚类过程，下面我们将着重介绍。

4.6.1 初始化

具体流程如4.6所示，1-6 行，我们计算每个元素到其他元素的距离，同时获取邻域关系存储在 u 中，并且记录每个元素邻域范围内包含点的个数 $s[j], j \in [n]$。8-9 行，我们并行比较 s 与 minpts 的大小关系，并更新每个元素的 isCore 属性。

算法 4.4 深度优先遍历 (dfs)

输入: 起始位置 idx，归属簇标识 mark，访问标识数组 visited，元素数组 plist，二维矩阵 matrix，映射关系 resmap

输出: 修改后的输入值

```
1: for i = 1 to n do
2:         if visited[i] 为真 then
3:                 该元素已经被划分好，不再继续
4:         end if
5:         if matrix[idx][i] > 0 then
6:                 resmap[plist[i].cluId] = mark
7:                 visited[i]=true
8:                 dfs(i, mark, visited, plist,matrix, resmap)
9:         end if
10: end for
```

算法 4.5 还原结果

输入: 明文 plist

输出: 聚类划分结果映射关系

```
1: 构造二维数组 matrix，大小为 n × n 存储相连关系
2: 构建 resmap 存储初始簇 id 到结果簇 id 的映射
3: for each p in plist do
4:         matrix[p.id1][p.id2] += p.isConn
5: end for
6: 构造访问标识数组 visited = [false]_n
7: mark = 1
8: for i = 1 to n do
9:         if visited[i] 为真 then
10:                不再继续访问
11:        end if
12:        visited[i] = true
13:        mark += 1
14:        resmap[plist[i].cluId] = mark
15:        执行 dfs(i, mark, visited, plist,matrix, resmap)
16: end for
```

10-20 行，我们开始算法的核心部分，即更新边界元素的 cluId。11 行，我们预设标识 m 来判断是否对元素更新 cluId，判断依据是若邻域范围内存在核心点，则可以修改（15 行）。13-14 行，若 $x[j]$ 为核心点且与 $x[i]$ 相邻，则 $t1[j]$ 为 1，否则 $t2[j]$ 为 1。根据该值，我们在 17 行，计算出 $r1$，过滤出所有相邻核心点的距离，

无关距离设为最大值，避免在 18 行影响求解最小值。19 行，我们通过逐个相乘后累加来求解最近相邻核心点的 cluId。在 20 行，我们根据前述 m 值来更新 cluId。

值得指出的是，对于核心点，$m=1$ 需要修改 cluId，但是最近相邻核心点为自身，因此 cluId 未改变。对于边界点，依据算法将修改 cluId 与最近相邻核心点保持一致。对于噪声，$m=0$ 因此 cluId 不会发生改变。

算法 4.6 初始化

输入: 秘密共享的元素数组 $\{x_1, ..., x_n\}$
输出: 距离关系 u

1: for $i = 1$ to n do
2: for $j = 1$ to n do
3: $d[j] \leftarrow \text{SED}(x[i].data, x[j].data)$
4: end for
5: $u[i] \leftarrow \text{SC}([eps]_n, d)$
6: $s[i] \leftarrow sum(u[j])$ for $j \in [n]$
7: end for
8: $r \leftarrow \text{SC}(u, [minpts]_n)$
9: $x[i].isCore \leftarrow r[i]$ for $i \in [n]$
10: for $i = 1$ to n do
11: $m \leftarrow 0$
12: for $j = 1$ to n do
13: $t1[j] \leftarrow \text{MUL}(x[j].isCore, u[i][j])$
14: $t2[j] \leftarrow 1 - t1[j]$
15: $m = t1[j] + m - \text{MUL}(m, t1[j])$
16: end for
17: $r1 \leftarrow \text{MUL}(d[i], t1) + \text{MUL}([max]_n, t2)$
18: $r2 \leftarrow \text{SMin}(r1)$
19: $nId \leftarrow \text{sum}(\text{MUL}(r2[j], x[j].cluId))$ for $j \in [n]$
20: $x[i].CluId \leftarrow \text{MUL}(nId, m) + \text{MUL}(1 - m, x[i].cluId)$
21: end for

4.6.2 聚类

4.7 基于 dbscan 的隐私保护层次聚类

4.8 理论分析

4.9 实验评估

算法 4.7 初始化

输入: 秘密共享的元素数组 $\{x_1, ..., x_n\}$，距离关系 u

输出: 相连关系 $plist$

```
1: for i = 1 to n do
2:         for j = 1 to n do
3:                 r ← MUL(x[i].isCore, x[j].isCore)
4:                 r ← MUL(r, u[i][j])
5:                 add new item to plist(x[i].cluId, x[j].isCore, r)
6:         end for
7: end for
```

4.10 本章小结

致　谢

衷心感谢导师 xxx 教授和 xxx 副教授对本人的精心指导。他们的言传身教将使我终生受益。

感谢 NUDTPAPER，它的存在让我的论文写作轻松自在了许多，让我的论文格式规整漂亮了许多。

参考文献

[1] Sadiku M N, Musa S M, Momoh O D. Cloud computing: opportunities and challenges[J]. IEEE potentials, 2014, 33(1): 34~36.

[2] Ribeiro M, Grolinger K, Capretz M A. Mlaas: Machine learning as a service[C]// 2015 IEEE 14th international conference on machine learning and applications (ICMLA). 2015: 896~902.

[3] Hunt T, Song C, Shokri R, et al. Chiron: Privacy-preserving machine learning as a service[J]. ArXiv preprint arXiv:1803.05961, 2018.

[4] Ahmed M, Seraj R, Islam S M S. The k-means algorithm: A comprehensive survey and performance evaluation[J]. Electronics, 2020, 9(8): 1295.

[5] Cramer R, Damgård I B, et al. Secure multiparty computation[M]. Cambridge University Press, 2015.

[6] Li P, Li J, Huang Z, et al. Privacy-preserving outsourced classification in cloud computing[J]. Cluster Computing, 2018, 21: 277~286.

[7] Kanungo T, Mount D M, Netanyahu N S, et al. An efficient k-means clustering algorithm: Analysis and implementation[J]. IEEE transactions on pattern analysis and machine intelligence, 2002, 24(7): 881~892.

[8] El Malki N, Ravat F, Teste O. KD-means: clustering method for massive data based on kd-tree[C]//22nd International Workshop On Design, Optimization, Languages and Analytical Processing of Big Data-DOLAP 2020-: vol. 2572. 2020.

[9] Yao A C C. How to generate and exchange secrets[C]//27th annual symposium on foundations of computer science (Sfcs 1986). 1986: 162~167.

[10] Rivest R L, Shamir A, Tauman Y. How to leak a secret[C]//Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. 2001: 552~565.

[11] Blakley G R. Safeguarding cryptographic keys[C]//Managing Requirements Knowledge, International Workshop on. 1979: 313~313.

[12] Beaver D. Efficient multiparty protocols using circuit randomization[C]// Advances in Cryptology—CRYPTO’91: Proceedings 11. 1992: 420~432.

[13] Schneider T, Zohner M. GMW vs. Yao? Efficient secure two-party computation with low depth circuits[C]//Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17. 2013: 275~292.

[14] Riazi M S, Weinert C, Tkachenko O, et al. Chameleon: A hybrid secure computation framework for machine learning applications[C]//Proceedings of the 2018 on Asia conference on computer and communications security. 2018: 707~721.

[15] Wu W, Liu J, Wang H, et al. Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 33(10): 3424~3437.

[16] Bunn P, Ostrovsky R. Secure two-party k-means clustering[C]//Proceedings of the 14th ACM conference on Computer and communications security. 2007: 486~497.

[17] Liu L, Su J, Liu X, et al. Toward highly secure yet efficient KNN classification scheme on outsourced cloud data[J]. IEEE Internet of Things Journal, 2019, 6(6): 9841~9852.

[18] Goldreich O. Encryption Schemes. The Foundations of Cryptography, vol. 2[Z]. 2004.

[19] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations[C]//Computer Security-ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings 13. 2008: 192~206.

[20] Rathee D, Rathee M, Kumar N, et al. CrypTFlow2: Practical 2-party secure inference[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 325~342.

[21] Mohassel P, Rosulek M, Trieu N. Practical privacy-preserving k-means clustering[J]. Cryptology ePrint Archive, 2019.

[22] Naeem M, Asghar S. Kegg metabolic reaction network data set[J]. The UCI KDD Archive, 2011.

[23] Fränti P, Sieranoja S. K-means properties on six clustering benchmark datasets[J]. Applied intelligence, 2018, 48: 4743~4759.

[24] Jäschke A, Armknecht F. Unsupervised machine learning on encrypted data[C]//Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers. 2019: 453~478.

[25] Rong H, Wang H, Liu J, et al. Privacy-Preserving-Means Clustering under Multiowner Setting in Distributed Cloud Environments[J]. Security and Communication Networks, 2017, 2017.

[26] Khan K, Rehman S U, Aziz K, et al. DBSCAN: Past, present and future[C]// The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014). 2014: 232~238.

[27] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise.[C]//Kdd: vol. 96: 34. 1996: 226~231.

[28] Tran T N, Drab K, Daszykowski M. Revised DBSCAN algorithm to cluster data with dense adjacent clusters[J]. Chemometrics and Intelligent Laboratory Systems, 2013, 120: 92~96.

[29] Latifi-Pakdehi A, Daneshpour N. DBHC: A DBSCAN-based hierarchical clustering algorithm[J]. Data & Knowledge Engineering, 2021, 135: 101922.

[30] Gheid Z, Challal Y. Efficient and privacy-preserving k-means clustering for big data mining[C]//2016 IEEE Trustcom/BigDataSE/ISPA. 2016: 791~798.

[31] Doganay M C, Pedersen T B, Saygin Y, et al. Distributed privacy preserving k-means clustering with additive secret sharing[C]//Proceedings of the 2008 international workshop on Privacy and anonymity in information society. 2008: 3~11.

[32] Yu T K, Lee D, Chang S M, et al. Multi-party k-means clustering with privacy consideration[C]//International symposium on parallel and distributed processing with applications. 2010: 200~207.

作者在学期间取得的学术成果

发表的学术论文

[1] Yang Y, Ren T L, Zhang L T, et al. Miniature microphone with silicon-based ferroelectric thin films. Integrated Ferroelectrics, 2003, 52: 229~235. (SCI 收录, 检索号:758FZ)

[2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究. 中国机械工程, 2005, 16(14): 1289~1291. (EI 收录, 检索号:0534931 2907)

[3] 杨轶, 张宁欣, 任天令, 等. 集成铁电器件中的关键工艺研究. 仪器仪表学报, 2003, 24(S4): 192~193. (EI 源刊)

[4] Yang Y, Ren T L, Zhu Y P, et al. PMUTs for handwriting recognition., In press. (已被 Integrated Ferroelectrics 录用. SCI 源刊.)

[5] Wu X M, Yang Y, Cai J, et al. Measurements of ferroelectric MEMS microphones. Integrated Ferroelectrics, 2005, 69: 417~429. (SCI 收录, 检索号:896KM.)

[6] 贾泽, 杨轶, 陈兢, 等. 用于压电和电容微麦克风的体硅腐蚀相关研究. 压电与声光, 2006, 28(1): 117~119. (EI 收录, 检索号:06129773469.)

[7] 伍晓明, 杨轶, 张宁欣, 等. 基于 MEMS 技术的集成铁电硅微麦克风. 中国集成电路, 2003, 53: 59~61.

研究成果

[1] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A. (中国专利公开号.)

[2] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102. (美国发明专利申请号.)

附录 A　模板提供的希腊字母命令列表

大写希腊字母：

Γ \Gamma	Λ \Lambda	Σ \Sigma	Ψ \Psi
Δ \Delta	Ξ \Xi	Υ \Upsilon	Ω \Omega
Θ \Theta	Π \Pi	Φ \Phi	
$\varGamma$ \varGamma	$\varLambda$ \varLambda	$\varSigma$ \varSigma	$\varPsi$ \varPsi
$\varDelta$ \varDelta	$\varXi$ \varXi	$\varUpsilon$ \varUpsilon	$\varOmega$ \varOmega
$\varTheta$ \varTheta	$\varPi$ \varPi	$\varPhi$ \varPhi	

小写希腊字母：

α \alpha	θ \theta	o o	τ \tau
β \beta	ϑ \vartheta	π \pi	υ \upsilon
γ \gamma	ι \iota	ϖ \varpi	ϕ \phi
δ \delta	κ \kappa	ρ \rho	φ \varphi
ϵ \epsilon	λ \lambda	ϱ \varrho	χ \chi
ε \varepsilon	μ \mu	σ \sigma	ψ \psi
ζ \zeta	ν \nu	ς \varsigma	ω \omega
η \eta	ξ \xi	$\varkappa$ \varkappa	$\digamma$ \digamma
$\upalpha$ \upalpha	$\uptheta$ \uptheta	o \mathrm{o}	$\uptau$ \uptau
$\upbeta$ \upbeta	$\upvartheta$ \upvartheta	$\uppi$ \uppi	$\upupsilon$ \upupsilon
$\upgamma$ \upgamma	$\upiota$ \upiota	$\upvarpi$ \upvarpi	$\upphi$ \upphi
$\updelta$ \updelta	$\upkappa$ \upkappa	$\uprho$ \uprho	$\upvarphi$ \upvarphi
$\upepsilon$ \upepsilon	$\uplambda$ \uplambda	$\upvarrho$ \upvarrho	$\upchi$ \upchi
$\upvarepsilon$ \upvarepsilon	$\upmu$ \upmu	$\upsigma$ \upsigma	$\uppsi$ \uppsi
$\upzeta$ \upzeta	$\upnu$ \upnu	$\upvarsigma$ \upvarsigma	$\upomega$ \upomega
$\upeta$ \upeta	$\upxi$ \upxi		

希腊字母属于数学符号类别，请用\bm命令加粗，其余向量、矩阵可用\mathbf。