分类号 TP391 学号 1605xxxx

UDC 681 密级 公开

工学硕士学位论文

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

学科专业 电子信息

研究方向 多媒体信息系统与虚拟现实技术

指导教师 付绍静 教授

国防科技大学研究生院

二〇二三年二月

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

Candidate: DengYanxiang

Supervisor: Prof. XX Zhang

A dissertation

Submitted in partial fulfillment of the requirements

for the degree of Master of Engineering

in Control Science and Engineering

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，除了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它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

国防科技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: **国防科技大学**; 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

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。好在，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 [5]。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 [6]。

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 [7]。

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 [8]，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 [9]，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。

3.2 预备知识

3.3 问题描述

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。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：

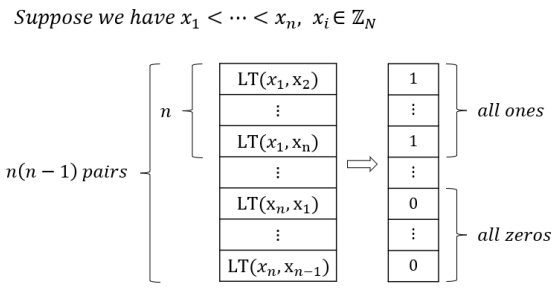

图 3.1　Secure minimal on small dataset (SMin(S))

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3.2所示。

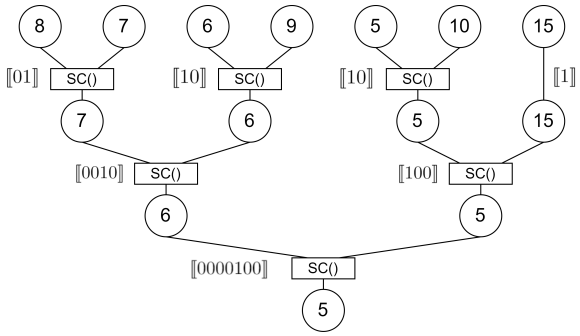

图 3.2　Secure minimal on large dataset (SMin(L))

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。

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

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: $\{\langle z'_1\rangle, ..., \langle z'_k\} \leftarrow \text{SF}(\langle z_1\rangle, ..., \langle z_k\rangle)$
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**

3.6 理论分析

3.7 实验评估

3.8 本章小结

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

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。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。

4.3 问题描述

4.3.1 系统模型

4.3.2 安全模型

4.3.3 设计目标

4.4 基于秘密共享的隐私保护计算模块

4.5 隐私保护 dbscan

4.6 改进的隐私保护 dbscan

4.7 基于 dbscan 的层次聚类

4.8 理论分析

4.9 实验评估

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。

作者在学期间取得的学术成果

公开评阅信息

序号	评阅人	职称	导师类型	工作单位	总分	结论	答辩建议	熟悉程度	备注
1	张三	教授	博导	XXX大学	95.8	达到	无需修改直接答辩	有深入了解	
2	李四	研究员	硕导	XXX大学	95	达到	修改后答辩	有深入了解	
3	王五	教授	博导						
4	赵六	教授	博导						
5	孙六	教授	博导	XXX大学	59	尚未达到	修改后复评	有深入了解	
	孙六	教授	博导	XXX大学	80	尚未达到	无需修改直接答辩	有深入了解	复评结果

说明：

1. 结论选项包括 2 个：“达到博士学位论文要求”、“尚未达到博士学位论文要求”。

2. 答辩建议选项包括 4 个：“无需修改直接答辩”、“修改后答辩”、“修改后复评”、“不予答辩”。

3. 熟悉程度选项包括 3 个：“有深入了解”、“比较熟悉”、“一般了解”。

提醒（正式成文后删除）：

1. 评阅版论文删除此页。

2. 采用双盲评阅方式的学位申请人撰写的学位论文删除此页。

3. 评阅总分无需取整。

4. 工作单位填至学校、科研院所即可。

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。