

学校代码： 10246  
学 号： 11210240037

# 復旦大學

## 硕士 学位 论文

### 图上可达性查询的隐私保护方法研究

Privacy-preserving Reachability Query Services

院 系： 计算机科学技术学院

专 业： 计算机软件与理论

姓 名： 尹树祥

指 导 教 师： 周水庚 教授

完 成 日 期： 2014 年 4 月 15 日

# 指导小组成员名单

周水庚 教授

# 目录

<b>摘要</b> . . . . .	1
<b>Abstract</b> . . . . .	2
<b>第一章 引言</b> . . . . .	3
1.1 研究背景和意义 . . . . .	3
1.2 研究内容和研究成果 . . . . .	4
1.3 本文结构 . . . . .	6
<b>第二章 相关工作</b> . . . . .	7
2.1 图数据库上隐私保护方法概述 . . . . .	7
2.2 图数据库查询验证 . . . . .	8
2.3 基于隐私保护的图查询服务 . . . . .	8
2.4 可达性查询算法概述 . . . . .	9
2.5 本章小结 . . . . .	13
<b>第三章 问题定义与系统概述</b> . . . . .	14
3.1 2-hop索引方法 . . . . .	14
3.2 密码学知识 . . . . .	16
3.3 可达性查询的隐私保护问题定义 . . . . .	17
3.4 本章小结 . . . . .	18
<b>第四章 图上可达性查询的隐私保护算法</b> . . . . .	19
4.1 具有Imax感知的2-hop算法 . . . . .	19
4.2 代理节点的添加算法 . . . . .	21
4.2.1 交集大小归一化 . . . . .	21
4.2.2 Lin和Lout大小归一化 . . . . .	25
4.3 索引加密处理算法 . . . . .	27

---

4.4 隐私保护的可达性查询处理 . . . . .	28
4.5 图上可达性查询的隐私保护算法的隐私分析 . . . . .	29
4.6 实验结果 . . . . .	32
4.6.1 合成数据集上的实验 . . . . .	33
4.6.2 真实数据集上的实验 . . . . .	35
4.7 本章小结 . . . . .	36
<b>第五章 稀疏图上可达性查询隐私保护方法 . . . . .</b>	<b>37</b>
5.1 m-2-hop索引算法 . . . . .	37
5.1.1 m-2-hop算法分析 . . . . .	38
5.1.2 基于启发式方法的m-2-hop构建算法 . . . . .	39
5.2 基于m-2-hop的可达性查询的隐私保护算法 . . . . .	41
5.3 实验结果 . . . . .	42
5.4 本章小结 . . . . .	43
<b>第六章 总结与展望 . . . . .</b>	<b>44</b>
<b>参考文献 . . . . .</b>	<b>45</b>
<b>学术论文 . . . . .</b>	<b>50</b>
<b>致谢 . . . . .</b>	<b>51</b>

## 摘要

在数据库领域中，越来越多的数据通过图结构的方式进行存储，比如社交网络、生物信息学以及XML数据等。针对这些图数据的查询，很多时候需要更强大的计算机资源进行处理。为解决该问题，比较经济实用的方法是将这些数据外包给一个第三方的服务提供商(Service Provider,  $\mathcal{SP}$ )，让它去为我们提供查询服务。然而， $\mathcal{SP}$ 并不是一直可以值得信赖，它可能对外包到它上面的数据比较感兴趣，以获得商业利益。除此之外，数据拥有者(Data Owner)不希望他的数据被不信任的人获取，同时查询者也不希望他的查询内容被他人得知。

可达性查询作为数据库领域一个最基础的问题，本文针对可达性查询过程中可能产生的隐私问题，提出了图数据上可达性查询的隐私保护方法。我们的隐私保护目标主要包括：客户端查询信息和图数据结构信息。针对我们提出的隐私保护目标，我们在基于传统的2-hop索引方法上，根据图数据不同的稀疏性质，提出了pp-2-hop和ppm-2-hop两种索引方法。在这两种索引方法中，我们对原始的2-hop索引通过不同的添加人工代理中心点策略，使得对于任何的可达性查询，其查询结果大小一致，进而 $\mathcal{SP}$ 无法从相同大小的查询结果中获取关于查询的信息。同时，我们通过对索引进行加密，使得所有的可达性查询在密文域中进行，进一步阻止 $\mathcal{SP}$ 对索引内容进行推理。本文对两种索引算法分别在基于密文攻击和基于集合大小的攻击模式下进行安全性分析。最后，分别对两种索引算法在合成数据集和真实数据集上进行实验研究。

**关键词：**图数据、可达性查询、2-hop索引、隐私保护、查询服务

**中图分类号：** TP311

# Abstract

Due to the massive volume of graph data from a wide range of recent applications, such as social network, bioinformatics and XML data. At the same time, much more efficient and powerful resources required to process numerous queries. It is becoming economically appealing to outsource graph data to a third-party service provider ( $\mathcal{SP}$ ) to provide query services. However,  $\mathcal{SP}$  cannot always be trusted. Hence, data owners and query clients may prefer not to expose their data graphs and queries.

Reachability query is one of the most fundamental query in database area. This paper studies privacy-preserving query services for reachability query where both clients' queries and the structural information of the owner's data are protected. According to different types of graph data, we propose two different index method base on 2-hop labeling, named privacy-preserving 2-hop labeling (pp-2-hop) and privacy-preserving minimum unified intesection 2-hop (ppm-2-hop) . In these two methods, the reachability queries are computed in an encrypted domain and the input and output sizes of any queries are indistinguishable. We analyze the security of these two method with respect to ciphertext only and size based attacks. We verify the performance of pp-2-hop and ppm-2-hop with an experimental study on both synthetic and real-world datasets.

**Keywords:** Graph data, Reachability Query, 2-hop, Index, Privacy Preserving Method, Query Service

**Classification Code:** TP311

# 第一章 引言

## 1.1 研究背景和意义

随着大数据的发展，越来越多的应用通过图结构的方式进行数据的存储，例如生物信息学、社交网络、通信网络、web拓扑结构以及半结构化数据XML等等。针对图数据的检索和挖掘，如今已经提出了诸多成熟算法。然而，随着图数据数量的爆炸式增长，提供一种快速高效的图查询服务成为了一项具有挑战性的任务。很多时候，这些图数据的拥有者并没有维护这些查询服务的专业知识，或者他们并不想自己去维护这样一个查询服务，他们更希望寻找拥有高性能集群或服务器的专业服务提供商( $\mathcal{SP}$ )为他们维护这样的服务。然而，安全和隐私保护作为服务质量(QoS)[34]的一个衡量指标， $\mathcal{SP}$ 并不是一直可以值得信赖的，由于这个因素的影响，作为数据的拥有者和查询服务的使用者，他们并不愿意去使用一个具有安全隐患的服务提供商。

可达性查询[17]是图数据库中一种基础的查询类型。其定义为：给定图上的两个顶点 $u$  和 $v$ ，可达性查询是指通过特定的算法检验顶点 $u$ 是否可以到达顶点 $v$ 。在这种应用场景下，查询者可能不希望将他的可达性查询内容（即两个顶点 $u$  和 $v$ ）被 $\mathcal{SP}$ 知道，另一方面，作为图数据的拥有者，也不希望他的数据被未经授权的第三方获取或者推理得到。因此，所有查询的结果都必须使用密文的方式进行加密，以使得不可信赖的 $\mathcal{SP}s$ 不能通过穷举或者基于知识推理的方法来还原整个图的结构信息，同时使得 $\mathcal{SP}s$  不能知道查询者的查询结果。

考虑图 1.1 中的一个例子，有一家制药公司，它主要的收入都来自于研发一

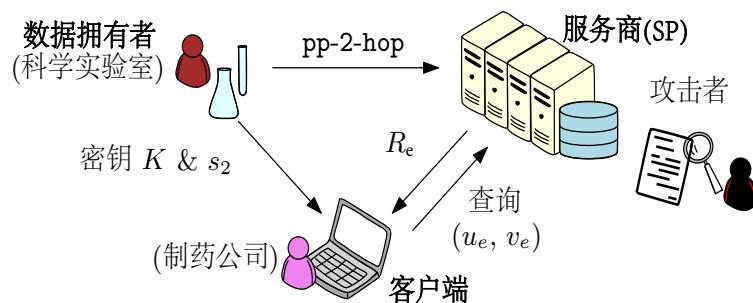


图 1.1: 系统模型概要

些保健产品。这家公司可能刚刚发现了一种新产品的制药配方。为了节省做实验的人力成本，它常常需要去一个非常庞大的公开生物信息网络(例如[4])中去查询这些混合物的信息，以了解这些混合物之间的相互作用关系(可能是通过判断两种混合物在这个生物信息的网络上是否通过一定的化学作用可以转化，这就是一种可达性查询)，判断这样一个混合物是否可以通过一定的生物酶的作用从另外一种或者几种化合物转化而来。然而从公司的角度来说，他们并不希望 $\mathcal{SP}$ 知道他们的查询内容，因为这可能涉及到知识产权问题，同时他们也不希望他们的竞争对手知道他们的查询内容。作为生物信息图数据的拥有者，他们一方面缺乏维护这样一个查询服务的经验，可能只有将数据通过一定的处理外包给一些专业的IT公司[24]或者一些云计算平台(例如亚马逊AWS)，利用这些平台庞大的处理查询请求的计算能力为他们提供查询服务；另外一方面，他们不希望将他们拥有的数据曝光给未授权的一些机构或者个人。作为一种折中的解决方案，他们可能只希望将数据的查询权限授权给一些购买了他们服务的用户。因此，防止 $\mathcal{SP}$ 获取客户端查询内容，或者通过推理得出图数据库的原始结构是一件非常具有挑战性的工作。

再考虑另外一个例子，在社交网络中，人与人可能通过电话的方式进行联系，我们可以用图中的一个顶点表示一个人，点与点之间的边表示人们有过电话的联系。考虑在一个刑事案件的侦破早期，警察可能要对一个犯罪嫌疑人进行调查，他们想知道都有谁和这个犯罪嫌疑人有联系，那么警察会向移动通信服务商提出查询请求。但是移动通信服务商可能不希望自己去维护这样的查询服务，所以他可能将他们的通信网络数据外包给一个查询服务提供商 $\mathcal{SP}$ 为其提供查询服务。但是，在外包给查询服务商的时候，由于这样的通信网络中有很多的敏感性的信息，所以移动通信服务商不希望查询服务提供商 $\mathcal{SP}$ 能够获得这个通信网络的结构信息。同时，警察也不希望他们的查询内容和查询的结果被 $\mathcal{SP}$ 知道，因为这很有可能会暴露他们的侦查工作。在这种情况下，提供一种隐私保护的可达性查询服务变得非常迫切。

## 1.2 研究内容和研究成果

在本文中，我们的主要研究问题是：在查询服务提供商 $\mathcal{SP}$ 并不能被安全信任的情况下，针对图数据的拥有者如何保护其外包给服务提供商的原始数据结构信息；针对查询服务的使用者，如何保护其发送给服务提供商的查询内容不被 $\mathcal{SP}$ 获取。根据我们已有的知识来看，目前还没有关于这一问题的相关研究。

当前在学术界有很多关于高效的图数据上可达性查询算法的研究成果[6, 7, 13, 15, 27, 28, 37, 38, 40, 42, 44, 11, 20, 45, 29]。在[26]中，作者将图上可达性查询的算法分为三类，分别为基于传递闭包压缩的索引算法、优化的在线搜

索查询算法和基于hop的索引算法。这三类算法中，第一类通常具有高昂的存储开销，对于部分大的图数据，这种方法可能无法工作；第二类算法由于需要在线的搜索，通常具有较慢的查询速度；第三类算法通常查询时间和索引大小介于以上两者之间。在第二章中我们将详细讨论各类算法的特点。在本文中，我们提出了基于2-hop的索引算法[17]对可达性查询进行隐私保护。采用2-hop的索引方法来实现隐私保护的可达性查询算法主要有以下三方面的优势：第一、2-hop索引的结构非常简单。在2-hop索引中，图中的每个顶点 $u$ 与其他顶点之间的可达性关系都通过两个集合( $\text{Lin}(u)$ 和 $\text{Lout}(u)$ )来表示。即在2-hop索引中，对于图中的每个顶点 $u$ 我们都使用两个集合 $\text{Lin}(u)$  和 $\text{Lout}(u)$ 来表示顶点 $u$ 与图中其他顶点之间的可达性关系。第二、基于2-hop 索引的可达性查询算法很精巧与简单，要判断两个顶点 $(u,v)$ 是否可达，我们只需要做一个简单的交集运算就可以判断两个顶点是否可达。正是这样简单的索引结构和精巧的可达性查询算法使得在做可达性查询的同时保护隐私成为了可能。第三、在学术界，2-hop索引方法仍然是一个非常热门的索引算法。在2-hop索引算法基础上，关于大图分割[15]、索引压缩算法[14]和索引更新算法[6, 37]这些方面的研究成果可以很容易的应用到我们的隐私保护可达性查询算法上。

在本文中，针对一般的图数据，在2-hop索引算法的基础上，我们提出了pp-2-hop(隐私保护的2-hop算法)。该算法主要思想是在2-hop的查询算法中，要判断两个顶点是否可达，仅仅需要在两个中心顶点集合之间做一个交集操作。基于以上的查询算法流程，在建立索引时，我们通过相应的启发式算法使得所有可能的两个集合之间的交集结果尽可能小，同时通过在每个顶点的 $\text{Lin}$  和 $\text{Lout}$ 集合中加入最少的人工代理中心点使得任何两个集合之间的交集大小相同。这一步的意义在于，对于任何查询，在 $\mathcal{SP}$ 看来，他们查询得到的交集大小完全一致，以避免不同交集大小带来的信息泄露。然后同时通过相应的算是使得建立出来的每个点的 $\text{Lin}$ ,  $\text{Lout}$ 的集合大小也在一个用户指定的差值范围内。在前面建立好的索引的基础上，我们再通过对索引进行加密操作，然后提出了基于索引密文的可达性查询算法，同时，我们对这样的一个隐私保护的2-hop索引方法进行隐私保护证明。

针对稀疏图，我们提出了另一种基于2-hop的索引算法ppm-2-hop，在该算法中，我们通过算法保证图中各个顶点之间的2-hop索引交集大小始终为1。这样做好处在于：对于稀疏图，图中只有极少的顶点之间是可达的，所以如果要使图中任意两个顶点的2-hop交集结果都为大于1的集合，我们需要引入较多的人工代理中心点来实现隐私保护的目的，但是如果使得所有的交集大小为1，可以在保证安全特性的前提下，大幅度的减少索引的大小。

本文中，我们的对于可达性查询的隐私保护问题的主要贡献可以总结为以下六点：

- 我们设计一种新的建立2-hop索引的启发式函数，使用该函数建立的2-hop索引能够保证任何两个顶点之间的索引交集大小较小，同时也能使得全局的2-hop索引大小最小。
- 我们提出了两种人工代理中心点添加算法，使得图中每个顶点的2-hop索引具有相同的大小，同时使得所有可能查询的查询结果具有相同的交集大小。
- 我们提出了一种可达性查询的隐私保护索引结构pp-2-hop，该索引可以保证原始图数据数据结构和可达性查询结果的安全性。
- 我们提出了在密文索引环境下的可达性查询算法，并对其优化，提出了一种基于乘法同态加密的优化查询算法。
- 针对稀疏图的特殊性，我们提出了另一种高效的隐私保护可达性索引方法ppm-2-hop。
- 从理论上和实验上，我们证明了两种可达性查询隐私保护算法的高效性和可扩展性。

### 1.3 本文结构

本文的组织结构如下：在第二章中我们介绍目前在图可达性查询领域的一些相关工作。在第三章中，将介绍本文算法的背景知识、问题定义和解决方法的概要介绍。在第四章中，我们提出了pp-2-hop索引建立算法，索引优化算法和查询流程优化，并通过实验对我们的索引方法及其查询效率进行验证。在第五章中，我们针对稀疏图提出了一种更加优化的ppm-2-hop索引方法，并通过实验，对该算法在稀疏图上的数据特性进行分析。在第六章中，我们对本文进行总结，并对将来可以进一步改进的方向进行展望。

## 第二章 相关工作

由于图数据的基础性和其应用的广泛性，图数据上的查询研究受到大家的广泛关注。而可达性查询作为图数据查询中最基础的算法之一，它的研究更是受到了研究员们的广泛兴趣。在本章节中，我们首先介绍关于图数据查询上目前一些比较流行的隐私保护研究，然后介绍当前一些比较成熟和优秀的图数据隐私保护查询的工作，最后我们再介绍当前在图数据可达性查询方面比较优秀的算法。

### 2.1 图数据库上隐私保护方法概述

图数据库一直是学术界和工业界广泛关注的最基本的数据结构，同时随着人们越来越注重隐私的保护，关于图数据隐私保护方面的成果非常多。它们中的大部分都是通过一定的算法将图的结果进行匿名化或模糊化，例如[12, 33, 47, 49, 5, 46, 18]。这些方法的主要思想为通过修改原始的图数据使得能够满足安全性要求。我们可以将目前比较先进的匿名化方法分为以下三类：

- 通过对图数据中边的信息进行修改的 $k$ -匿名( $k$ -anonymity)隐私保护方法。在该方法中，通过对图中的一系列边的删除与添加，使得修改后的图中的每一个点都至少有 $k-1$ 个点和其具有相同的结构形式。
- 图数据边的随机化。在该类方法中，通过随机的添加、删除或者交换两条边使得攻击者无法通过基于概率的方法对原来的图数据结构进行猜测。
- 基于聚类的泛化方法。在该类方法中，图中的顶点和边被聚集成不同的分组，然后将一个子图转化成一个点。通过此类方法，图中的每一个单独的点的信息被隐藏。

例如，在[12]文章中，作者通过将原图文件划分为 $k$ 个不相交的同构子图，然后这样划分以后，虽然保护了隐私数据，但是基于划分的子图做可达性查询变成了一件几乎不可能完成的任务。在[33]文章中，作者指出现实世界里的图数据中顶点的度的信息是严重分布不均匀的，这使得攻击者很容易能够收集到一

个目标顶点的度的信息。他们研究了如何通过添加或者删除一个图中的部分边来获得一个k-degree的匿名化图数据。在处理后的图数据中，每个顶点都至少拥有k-1个和它具有相同度数的顶点。在该方法中，k-degree匿名化可以避免攻击者根据一些先验知识来对图中部分点的信息进行推测。虽然以上的三类方法能够有效的保护图数据的隐私信息，但是通过对图中顶点或者边的增减操作来实现图数据的模糊化，这些模糊化后的图数据改变了原图数据的特性，使得这些图数据不具有了与可达性查询相应的信息。目前，关于如何使用匿名图数据文件来完成指定的一些查询，在学术界和工业界都还没有这样的一些工作。

## 2.2 图数据库查询验证

查询验证是一个安全领域里的问题，通常在这种情况里面，服务提供商 $\mathcal{SP}$ 被认为是不安全和不可信赖的[31, 32, 8]。它要求客户端验证从服务器端返回的图数据的正确性和未被修改性。Kundu等人在[31, 32]文章中，他们主要研究了如何在不泄露数据(可能是树、图或是森林)任何额外的信息的情况下，验证用户可以从服务商获取到的部分数据(可能是原先数据的一个子图或者子树)的正确性，以防止服务商用一些非来自原始数据的虚假数据来欺骗用户。在他们的技术中，他们优化了验证数据的签名技术。在[21]文章中，作者提出了一种基于外包的图数据系统框架上的高效子图查询验证算法。在文章中，作者通过使用相应的技术，对从服务商返回的查询结果进行验证，进而保证用户查询结果的正确性。在本文中，我们主要的研究问题是如何在基于外包图数据框架下图数据上可达性查询技术的研究，在我们的问题中，我们主要注重如何通过相应的算法和框架使得我们在不泄露原始图结构信息和查询结果的情况下，高效的完成图上点之间的可达性查询。

## 2.3 基于隐私保护的图查询服务

在[23]文章中，作者分析了如何在保护图数据边信息的情况下，如何判断图上两点之间的可达性。但是他们的方法向服务商 $\mathcal{SP}$ 暴露的查询点和图数据的部分结构信息，无法做到真正的达到我们所定义的隐私保护级别。在[22]文章中，作者提出了一种云计算环境下基于保护临近节点信息的最短距离查询算法。在他们的文章中，他们希望在基于外包的图数据条件下能够实现保护邻居节点的信息的情况下，同时能够保护邻居节点间的连接关系和图上任意两点间的最短路信息。但是如果我们将他们的工作应用到图数据上可达性查询，关于图数据的一些结构信息和查询点之间的可达性信息仍然将会暴露给服务商 $\mathcal{SP}$ .

在[36]文章中，作者使用了PIR(Private information retrieval)，所谓PIR协议是指通过该协议可以使得用户从存储数据库中的服务器上检索一个条目，同时使得该服务器无法判断哪一个条目被用户检索过)[16]使得可以在计算两个查询点之间最短路径时没有任何关于图数据信息的泄露。但是，众所周知，PIR具有高昂的计算成本，并且使用PIR方法对于每个查询，为了实现隐私保护，需要传输相同数量的文件大小，这个传输数据的大小等同于所有查询中需要查询数据量最大的查询的数据量，那么数据传输成本也是非常昂贵。由于可达性查询是一个简单并且非常基础的查询，高昂的计算成本和传输代价使得PIR不适合用于可达性的隐私保护。在[9]文章中，作者提出了一种基于加密的图数据上的子图查询算法。在该文中，作者的方法提供了查询的隐私保护、索引的隐私保护以及图文件的特征隐私保护等功能，但是如果将其方法考虑到可达性查询上，可达性查询无法将一个查询变为一个子图查询。在[30]文章中，作者提出了一种基于将图数据的部分统计信息外包给服务商 $\mathcal{SP}$ 以保护图数据边微分隐私保护的算法。虽然他们的算法支持很多的计数以及查询算法，但是对于可达性查询，该算法无法实现。

## 2.4 可达性查询算法概述

可达性查询作为图论中最基础的查询之一，一个图 $G$ 上的可达性查询：查询顶点 $u$ 是否可以到达顶点 $v$ 。有两种最简单最直接的方法来处理这样一个可达性查询。第一个方法可以使用深度优先或者广度优先遍历方法从顶点 $u$ 出发，看能否遍历到顶点 $v$ 。使用这种搜索算法最坏的情况需要 $O(n + m)$ 的时间复杂度，其中 $n$ 和 $m$ 分别表示图 $G$ 的顶点数和边数。第二个方法，我们可以事先计算好图 $G$ 的传递闭包 $TC$ ，并将该传递闭包集合存储在硬盘上，在做查询时，只要检查元素 $(u, v)$ 是否在传递闭包集合中。在这种情况下，它需要 $O(n^2)$ 的存储开销。然而对于较大的图数据时，这两种最简单和最直接的方法都是不可行的，在前一种方法中需要太长的查询时间，在第二种方法中需要太多的存储空间来保存传递闭包集合，尤其当图数据具有百万个以上节点时，这些方法几乎无法实现可达性查询。

在目前的数据库研究领域，研究工作者们提出了很多的算法来降低存储开销的同时保持高效的可达性查询。正如前面介绍的一样，如果我们可以预先计算好图的传递闭包，我们可以用 $O(n^2)$ 存储开销换来 $O(1)$ 的查询时间，或者我们可以直接使用深度优先或者广度优先遍历算法，在不需要存储空间的情况下，可以在 $O(n + m)$ 时间里回答一个可达性查询。目前的研究工作就是在这两种极端情况里寻找一个平衡点，使得一个查询的查询时间在 $O(1)$ 到 $O(n + m)$ 之间，同时也可能需要较小的索引存储开销。例如，一些方法基于图 $G$ 的生成树并再加

上一些额外的信息来保存图 $G$ 的可达性信息来构建图的可达性索引。在另一些方法中，通过对传递闭包的压缩来获得一个大小可以接受的索引。基于这个方向的努力，构建索引的时间开销也变成了一种重要的考虑指标。

表 2.1: 各种主流的可达性索引时间和空间复杂度

索引方法	查询时间	索引创建时间	索引大小
Transitive Closure[39]	$O(1)$	$O(nm)$	$O(n^2)$
Tree+SSPI[10]	$O(m - n)$	$O(n + m)$	$O(n + m)$
GRIPP[41]	$O(m - n)$	$O(n + m)$	$O(n + m)$
Dual-Labelng[43]	$O(m - n)$	$O(n + m + t^3)$	$O(n + t^2)$
Tree Cover[3]	$O(\log(n))$	$O(nm)$	$O(n^2)$
Chain Cover[11]	$O(\log(k))$	$O(n^2 + kn\sqrt{k})$	$O(nk)$
Path-tree Cover[27]	$O(\log^2(k'))$	$O(nm)$	$O(nk')$
2-hop Cover[17]	$O(m^{1/2})$	$O(n^3 *  TC )$	$O(nm^{1/2})$
3-hop Cover[28]	$O(\log n + k)$	$O(kn^2 *  Con(G) )$	$O(nk)$

表 2.1给出了目前比较高效和流行的可达性查询索引的时间和空间复杂度。对于一个图 $G = (V, E)$ ，其中 $n = |V|$ ,  $m = |E|$ 。在[39]中作者提出了一种计算图的传递闭包的算法，该算法可以在 $O(nm)$ 内计算出一个图的传递闭包集合。换句话来说，计算一个基于传递闭包的可达性索引的时间复杂度为 $O(nm)$ ，而在最坏的情况下该索引的空间大小为 $O(n^2)$ 。基于该传递闭包索引，一个可达性查询的查询时间为常数时间 $O(1)$ 。

在[10]文章中，作者基于图 $G$ 的生成树提出了一种可达性索引方法。该索引构建算法的时间复杂度为 $O(n + m)$ ，索引的空间大小为 $O(n + m)$ 。给定图 $G$ 中的两个顶点 $u$  和 $v$ ，查询顶点 $u$ 是否可以到达顶点 $v$ 。如果图 $G$ 的生成树里有一条从 $u$ 到 $v$ 的路径，那么该算法可以在 $O(1)$ 时间里返回查询结果。对于如何快速判断生成树中是否存在这样一条路径，我们可以使用生成树上每一个顶点的编码通过一个简单的断言来实现，我们将该断言表示为 $P(,)$ 。我们假设在生成树上顶点 $u$ 和 $v$ 的编码分别是 $code(u)$ 和 $code(v)$ ，如果 $P(code(u), code(v))$ 为真，那么顶点 $u$ 可以到达顶点 $v$ 。由于生成树中每个顶点的编码是根据生成树上顶点之间的连接关系来确定的，如果 $P(code(u), code(v))$  为假并不代表顶点 $u$ 不可以到达顶点 $v$ ，因为图 $G$ 有一些其他的边并没有出现在生成树中。对于这种情况，作者使用了一个额外的数据结构，称之为SSPI(Surrogate and Surplus Predecessor Index)索引来在运行时来计算顶点之间的可达性信息。使用SSPI索引进行可达性查询在最坏的情况下时间复杂度为 $O(m - n)$ 。我们称这种可达性索引方法为Tree+SSPI。和[10]中类似，在[41]文章中，作者同样使用了一棵生成树来进行可达性的查询。在[41]文章中，作者基于生成树建立了GRIPP(GRaph Indexing

base on Preorder and Postorder numbering)索引。对于生成树上可以直接回答的可达性查询，可以在 $O(1)$ 时间给出结果。对于生成树中无法确定的查询，作者给出了一些基于GRIPP索引的遍历策略。在GRIPP中，算法的时间复杂度和空间复杂度和Tree+SSPI一致。

在[43]文章中，作者发现现实世界中，大部分应用中的大图都比较稀疏，所以作者提出了一种针对稀疏大图的dual-labeling索引方法。稀疏图意味着图 $G$ 中没有出现在它的一个生成树中的边非常少。我们假设没有出现在生成树中的边的数目为 $t$ ，很显然对于稀疏图 $t \ll n$ ，作者考虑为图中出现在生成树中的边建立一个树编码索引，对于图中未出现在生成树中的边，使用一种图编码的方式来建立索引，作者为图中未出现在生成树中的边计算它们的传递闭包集合。Dual-labeling可以实现 $O(1)$ 的可达性查询时间复杂度，需要的索引大小为 $O(n + t^2)$ ，索引的创建时间为 $O(n + m + t^3)$ 。

在[3]文章中，作者研究了一种树覆盖(Tree Cover)的索引方法来对图中的顶点进行索引。简单来说，如果一个顶点 $u$ 可以到达顶点 $v$ ，那么顶点 $u$ 可以到达以顶点 $v$ 为根的子树上的任何顶点。文章中，作者提出了一种最优的树覆盖的方法来最大程度的压缩图的传递闭包元素。在最坏的情况下，该索引的空间大小为 $O(n^2)$ ，然而在实际情况中，该索引方法通常比[11]中的链覆盖方法具有更好的结果。Tree Cover索引方法对大图建立索引非常高效，它的索引创建时间复杂度为 $O(nm)$ ，在Tree Cover索引上，可达性查询的时间复杂度为 $O(\log n)$ 。

在[25]文章中，作者提出了一种链覆盖(Chain Cover)的索引方法。Chain Cover的主要思想是将一个图 $G$ 分解成许多两两不相交的链。链的概念要比路径的概念更加概要。考虑图 $G$ 中的一条路径 $a \rightarrow b \rightarrow c \rightarrow d$ ，其中 $x \rightarrow y$ 表示图中的一条有向边。图中的一条路径可以被看成一条链， $a \rightsquigarrow b \rightsquigarrow c \rightsquigarrow d$ ，其中 $x \rightsquigarrow y$ 表示顶点 $x$ 可以到达顶点 $y$ 。图中的一条路径可以被分解成两对不相交的链，例如 $a \rightsquigarrow c$ 和 $b \rightsquigarrow d$ ，这里 $a \rightsquigarrow c$ 和 $b \rightsquigarrow d$ 都不是一条路径。和Tree Cover中一样，如果顶点 $u$ 可以到达顶点 $v$ ，那么顶点 $u$ 可以到达任何以顶点 $v$ 作为起点的链上的所有顶点。在文中，作者提出了一种时间复杂度在 $O(n^3)$ 的算法来找到一个图的最少数量的链覆盖。一个图 $G$ 中找到的链的数量我们称之为图 $G$ 的宽度，表示为 $k$ 。基于Chain Cover，我们可以构建出一个大小为 $O(nk)$ 的可达性索引，可达性查询的时间复杂度为 $O(\log k)$ 。在[11]中作者进一步提出了一种基于Chain Cover的优化索引方法，该索引创建时间复杂度可以降到 $O(n^2 + kn\sqrt{k})$ 。

在[27]文章中，作者延续Tree Cover中的思想，提出了一种path-tree cover的索引方法。作者通过将图 $G$ 分解成很多互不相交的路径，然后将分解后得到的每一个路径作为树中的一个顶点，建立一个tree cover索引。我们假设图分解得到的互不相交的路径数记为 $k'$ ，作者提出了两种算法构建可达性索引，分别称之为PTree-1和PTree-2。这两种算法构建的索引大小都为 $O(nk')$ 。算法PTree-1构

建可达性索引的时间复杂度为 $O(nm)$ ，算法PTree-2的时间复杂度为 $O(mk')$ 。基于这两种算法得到的可达性索引的查询时间复杂度都为 $O(\log^2 k')$ 。

在[17]中，作者提出了一种2-hop cover索引方法。对于图 $G$ 中的每个顶点 $u$ 都分配有两个集合( $\text{Lin}(u)$ 和 $\text{Lout}(u)$ )，这两个集合作为该顶点的索引。集合 $\text{Lin}(u)$ 中包含有一系列可以到达顶点 $u$ 的顶点，集合 $\text{Lout}(u)$ 中包含顶点 $u$ 可以到达的顶点集合。每个顶点的索引保证如果在图 $G$ 中，顶点 $u$ 可以到达顶点 $v$ ，则 $\text{Lout}(u) \cap \text{Lin}(v) \neq \emptyset$ 。找到一个图的2-hop cover被证明是一个集合覆盖问题。在文章中，作者提出了一种近似方法可以得到 $O(nm^{1/2})$ 大小的索引。该索引的时间复杂度还没有得到一个确定的值。在[29]中，作者给出了一个时间复杂度为 $O(n^3 * |TC|)$ ，其中 $|TC|$ 为图 $G$ 的传递闭包元素集合的大小。关于2-hop索引的构建方法我们将在第3.1中详细介绍。在[28]中，作者进一步提出了一种结合Chain Cover和2-hop的索引方法，我们称之为3-hop。3-hop索引算法的时间复杂度为 $O(kn^2 |Con(G)|)$ ，这里 $k$ 是图 $G$ 中分解得到的互不相交的路径的数目， $Con(G)$ 是文章中定义的关于传递闭包元素的一个参数。

以上为目前关于图数据上的可达性计算的一些典型的索引方法的介绍，总的来说，我们根据[26]文章中的分类，我们可以把目前比较流行的可达性查询算法分为基于传递闭包压缩的索引方法、对在线搜索优化的方法和基于hop的索引方法。

**传递闭包压缩算法** 对于这一类可达性查询索引方法的主要思想是对给定图数据的传递闭包进行压缩，减小索引对存储空间的需求，并且对图中的每个顶点都用一个可达的顶点集合进行编码。使用了这种方法进行索引以后，如果想要查询两个点 $u$ 和 $v$ 是否可达，我们仅仅需要查询顶点 $u$ 的可达性集合里是否包含顶点 $v$ ，如果包含则说明顶点 $u$ 可以到达顶点 $v$ ，反之则不可达。[3, 27, 42]等文章是目前比较先进高效的基于传递闭包的压缩算法。使用传递闭包压缩算法，他们的查询效率一般是最高效的，然而，此类方法的索引存储开销通常是最大的，有时候甚至是不可能实现的。

**优化的在线搜索算法** 这一类算法最基本的思想是利用在线搜索算法(如深度优先搜索算法DFS，广度优先搜索算法BFS)在线的计算两点之间的可达性。具体来说，他们首先利用一些额外的信息(例如通过后续拓扑遍历或者生成树)来对给定的查询进行搜索空间的剪枝。然后，再利用DFS或BFS搜索算法结合标签信息来得到最后的答案信息。实践证明，[38, 44]等算法可以支持具有千万级别的图数据上的可达性查询。然而，通常这些在线搜索算法(DFS或BFS)在进行检索时都携带有图数据的结构信息。目前来看，还不知道如何使得在进行深度优先搜索或广度优先搜索的同时能够保护图数据的结构信息。

**基于hop的索引算法** 基于hop的索引算法是目前图数据上可达性查询算法中最为流行的一类索引算法，该类算法通常不需要像传递闭包那么大的索引空间同时也比在线搜索要快。基于hop的索引算法最先提出的是[17]中的2-hop索引。在2-hop算法中，图中每个顶点的可达性信息都用两个集合，我们称之为Lin和Lout来进行标识。这些所有的Lin和Lout构成了整个图数据的传递闭包集合。为了判断两个点 $u$ 是否可以到达 $v$ ，我们只需要检查 $\text{Lout}(u)$ 和 $\text{Lin}(v)$ 的交集是否为空即可，若不为空，则说明顶点 $u$ 可以到达顶点 $v$ ，若为空，则说明不可达。为了进一步提高2-hop的查询和索引效率，很多基于2-hop的延伸工作[7, 13, 28]来提高原始的2-hop的索引和查询效率。同时，先前也有很多研究工作[14, 15, 37]来优化2-hop的索引大小以及它的高昂的计算成本，使得2hop能够支持更大的图，生成索引效率越高也降低最后的索引的大小。[13] 文章中，作者提出了一个类似于2-hop的索引方法，该方法大幅度的提升了查询的效率，降低了索引的大小同时索引的开销也是变得可以接受。同时，也有一些工作[6]注重于如何在建立好的索引基础上，高效的完成增量维护。

由于2-hop具有非常简单的索引结构，同时基于2-hop的可达性查询方法也仅仅只有一个交集计算。在本文中，我们基于2-hop来完成我们基于隐私保护的可达性查询算法。很多基于2-hop的延伸算法可以很简单的应用到我们的系统中。

## 2.5 本章小结

在本章中，我们简要的介绍了目前在图数据库领域中，在隐私保护以及图的可达性查询算法方面的一些文章。并对各种算法进行了简要的分析，总结了各种算法优点与缺点，并考虑了如果将各种算法应用到我们研究问题上所难以解决的问题。同时，我们对目前图中最基础的可达性查询算法进行简要的分类，总结每一类算法的特点，最后我们根据各类算法的特点，简要说明了在本文中，我们基于2-hop算法进行研究的好处与意义。

## 第三章 问题定义与系统概述

在本章中，我们将介绍本文研究的背景知识、问题的定义以及我们解决方案的一个概述。表 3.1 中列举出了本文中的常用符号。

### 3.1 2-hop索引方法

在本文中，我们考虑的是基于有向无环图的可达性查询。我们通常用 $G$ 来表示一个图数据， $V(G)$  和 $E(G)$  分别表示图的顶点集合和边的集合。由于在一个强连通分量里的各个点的可达性信息是完全一样的，所以在本文中，为了描述的方便性，我们假设所有的图为有向无环图(Directed Acyclic Graph, DAG)。对于图上的每一个点 $u \in V(G)$  都具有两个顶点集合 $\text{Lout}(u)$ 和 $\text{Lin}(u)$ ，我们把这两个集合称之为顶点 $u$ 的2-hop索引。对于 $\text{Lout}(u)$ 集合中的点，表示可以从顶点 $u$ 到达的点。对于 $\text{Lin}(u)$ 集合中的点，表示这些点可以到达顶点 $u$ 。这些 $\text{Lin}(u)$ 和 $\text{Lout}(u)$ 集合中的点，我们称之为中心点。对于给定两个点 $u$ 和 $v$ ，做可达性查询 $\text{Reach}(u, v)$ ，我们称顶点 $u$ 可以到达 $v$  当且仅当 $\text{Lout}(u) \cap \text{Lin}(v) \neq \emptyset$ ，可以记作为 $u \rightsquigarrow v$ ；否则如果 $\text{Lout}(u) \cap \text{Lin}(v) = \emptyset$ ，我们称顶点 $u$ 不可以到达顶点 $v$ ，我们记作为 $u \not\rightsquigarrow v$ 。为了保证图的顶点集合 $v \in V(G)$ 中所有顶点的2-hop索引信息能够包含图 $G$ 上顶点间所有的可达性信息，所以2-hop索引里包含的可达性信息必须包含图 $G$  传递闭包集合 $T(G)$  里面的所有元素。很明显，我们可以有很多种算法来计算图 $G$  的2-hop索引。之前的大部分工作都希望建立出来的2-hop索引大小能够最小，索引的大小我们可以定义为 $\sum_{u \in V(G)}(|\text{Lout}(u)| + |\text{Lin}(u)|)$ 。

2-hop的索引创建过程通常被认为非常耗时和对系统要求很高。尽管已经有很多工作(例如[15])明显的优化了原始的2-hop的创建时间，但是实际上，2-hop的创建基本上都是离线建立的。由于生物信息网络、通信网络等等数据更新不是非常的频繁，所以这些数据的拥有者可以先在本地建立好相应图数据的2-hop索引，然后再将处理完的2-hop索引上传到服务商 $\mathcal{SP}$ ，使得可以支持高并发的可达性查询请求。

我们在这里先简要介绍一下[17]文章中建立2-hop的算法流程，我们在后面将针对我们的问题，提出针对这种原始的算法的改进算法。首先，我们用一个

表 3.1: 文中常用符号

符号	描述
$\text{Reach}(u, v)$	一个可达性查询, 查询 $u$ 是否可以到达 $v$
$u \rightsquigarrow v$	图中的顶点 $u$ 可以到达顶点 $v$
$u \not\rightsquigarrow v$	图中的顶点 $u$ 不可以到达顶点 $v$
$\text{Lout}(u), \text{Lin}(u)$	顶点 $u$ 的2-hop 索引
$I_{\max}$	$I_{\max} = \forall u, v, \max( \text{Lout}(u) \cap \text{Lin}(v) )$
$\text{Lout}^s(u), \text{Lin}^s(u)$	顶点 $u$ 添加了人工点之后的2-hop 索引
$\text{Lout}_{\max} (\text{Lin}_{\max})$	$\text{Lout}^s (\text{Lin}^s)$ 中最大的集合的大小
$h_s(\cdot)$	带有参数 $s$ 的hash函数
$E(\cdot), K$	Elgamal加密算法, 解密的密钥 $K$
$u_e \rightsquigarrow v_e$	一个加密的可达性查询 $u_e = h_{s_2}(u), v_e = h_{s_2}(v)$
$\text{Lout}^e(u), \text{Lin}^e(u)$	顶点 $u$ 的加密的pp-2-hop
$w_e$	加密的中心点 $w, w_e = h_{s_1}(w)$
$f_e$	加密的标志位标志 $f, f_e = E(f)$
$R$	$\text{Lin}^e(u_e) \cap \text{Lout}^e(v_e)$ 的交集结果
$R_e$	服务端返回的加密了的可达性查询结果

变量 $T'$ 表示图 $G$ 的传递闭包 $T(G)$ 所有没有被覆盖到的元素, 也即 $T' = T(G)$ 。通过不断的迭代, 使得 $T(G)$ 中的元素不断的被覆盖并从 $T'$ 中移除。当集合 $T'$ 中的元素变为空时, 以上的迭代结束。对于图中的每一个点 $w \in G$ , 都对其建立一个相应的二分无向图(也可称为中心图) $G_w(L_w, R_w, E_w)$ , 这里的 $L_w$ 是指所有的可以到到点 $w$ 的点的集合,  $R_w$ 是所有点 $w$ 可以到的点的集合。 $(u, v) \in E_w$  当且仅当 $(u, v)$  在集合 $T'$ 中。然后启发式算法每次选择一个中心点 $w$ , 该中心点具有它对应的二分图 $G_w$ 中最大的maxDensCover比例的子图 $G_i(L_i, R_i, E_i)$ 。一个子图 $G_i(L_i, R_i, E_i)$ 的maxDensCover具体定义为:

$$\text{maxDensCover} = \frac{|E_i \cap T'|}{|L_i \cup R_i|} \quad (3.1)$$

这个问题实际上就变为查找图 $G_w$ 的最大稠密子图。在每一次的循环中, 具有最大的maxDensCover的点 $w$ 被选为一个中心点。对于所有的 $u \in L_i$  和 $v \in R_i$ , 算法将中心点 $w$ 加入到 $\text{Lout}(u)$ 和 $\text{Lin}(v)$ 中, 并从 $T'$ 中移除 $(u, w), (w, v)$ 和 $(u, v)$ 。通过以上的maxDensCover启发式方法, 每次选择一个中心点, 直到图 $G$ 的所有传递闭包元素 $T(G)$ 被覆盖完, 算法结束。

我们考虑图 3.1中左边的这样一个代表生物网络的简单有向无环图。图中, 顶点的标号表示一些化学物质的编号, 图中的边表示两种物质之间的一种化学反应。图中所有的顶点的标签为了叙述的简洁性, 我们在此略去。左图所对应

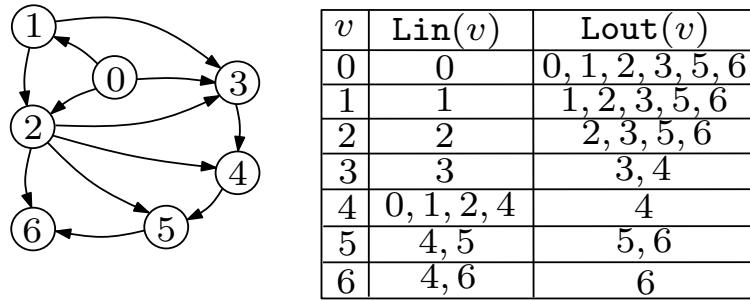


图 3.1: 图以及它的2-hop索引

的原始的2-hop索引文件见图 3.1右边表格。我们考虑查询顶点1可不可以到达顶点5，也即查询 $\text{Reach}(1, 5)$ ，我们可以看到 $\text{Lout}(1) \cap \text{Lin}(5) = \{5\}$ ，所以我们可以得到顶点1可以到达顶点5。然而对于顶点6到顶点0，由于 $\text{Lout}(6) \cap \text{Lin}(0) = \emptyset$ ，所以我们说顶点6无法到达顶点0。在这里我们用一个简单的例子说明了2-hop索引是如何建立的，同时如何利用建立的2-hop索引进行可达性的查询。

## 3.2 密码学知识

我们在第一章中介绍到，我们需要在加密的2-hop索引文件上进行可达性查询的处理。在本篇文章中，我们主要利用到以下两个密码学中的技术。

哈希函数就是把任意长度的输入，通过内部的散列算法，变换成固定长度的输出，这个输出就是散列值。这种转换是一种压缩映射，简单来说哈希函数是一种将任意长度的消息压缩到某一固定长度的消息摘要函数。通常一个哈希函数记为 $\text{hash}()$ 。对于一个给定的数据值 $m$ ，它的哈希值记为 $\text{hash}(m)$ 。哈希函数具有以下两个特性：(1)单向性，也即对于一个给定的哈希值 $\text{hash}(m)$ ，我们很难从这个哈希值推算出它原来的值。(2)抗冲突性，即通常对于不同的输入所产生的哈希值是一一对应的，很难找到两个不同的输入使得他们的哈希值一样。目前一种比较成熟和安全的哈希函数是SHA-1[35]。

在密码学领域中，通常在哈希函数中通过引入一个参数，称之为盐，来提高哈希函数的随机性。通常带有一个参数的哈希函数记为 $\text{h}_s()$ 。通过给定不同的盐参数，那么同样的哈希函数在相同的原始输入下所产生的输出是不同的。总结来说，如果 $s_1 \neq s_2$ ，那么 $\text{h}_{s_1}(m) \neq \text{h}_{s_2}(m)$ ，反之如果 $s_1 = s_2$ ，则 $\text{h}_{s_1}(m) = \text{h}_{s_2}(m)$ 。在本文中，我们使用了两个不同的盐参数来作为数据拥有着的私有密钥的一部分。

所谓乘法同态加密算法是指在它密文上进行的算法操作与在它明文上的乘法操作是同态的。一个乘法同态加密算法通常记为 $\text{E}(\cdot)$ ，对于乘法同态的定义具体来说，给定两个密文 $\text{E}(m_1)$ 和 $\text{E}(m_2)$ ，我们可以直接在密文上进行乘法计

算，它的结果等同于 $E(m_1 \times m_2)$ ，那么我们对乘法后的密文用密钥 $K$ 进行一次解密，那么就可以等到 $m_1 \times m_2$ 的值。在乘法同态加密领域，目前比较流行的加密算法是Elgamal算法[19]。在这里值得一提的是，由于Elgamal算法在加密的时候引入了随机因子，所以Elgamal保证了加密后的结果的随机性，简单来说，对于同样的数据用Elgamal加密算法加密完的结果是不一样的。例如 $m_1 = m_2$ ，但是 $E(m_1) \neq E(m_2)$ 。这一类加密算法在明文的域比较小的时候，作用就更加明显，可以通过算法中引入的随机因子，使得由明文较小的域变成密文后的域变大，防止通过基于频率的方法来对密文进行破解。

### 3.3 可达性查询的隐私保护问题定义

在本节中，我们着重从数据模型、系统模型、隐私保护目标以及攻击模型等四个方面来对本文的研究问题进行定义。

**数据模型** 我们首先给我本文中关于数据模型的定义。在本文中，我们考虑的是带有结点标签的有向图。我们用 $G$ 表示图数据库中的一个图数据，分别用 $V(G)$  和 $E(G)$  表示图的顶点集合和边的集合。由于在强连通分量里的所有点与其他点的可达性完全一致，所以为了在本文中叙述的简洁性，我们假设本文的讨论是基于有向无环图进行讨论，我们可以将所有的强连通分量转化为图中的一个点，在完成去掉强连通分量的图的索引建立后，我们可以通过很简单的方法，将强连通分量里每个点的可达性信息表达出来。一个可达性查询是指，输入两个查询点 $u$  和 $v$ ，可以表示为 $\text{Reach}(u,v)$ ，当 $u$ 可以到达顶点 $v$ 时，返回true。

**系统模型** 在本文中，我们遵循在数据库外包领域中最常见的系统模型来作为本文的系统模型的基础，这个模型见图 1.1。这个模型主要有三个重要组成部分。

- 数据拥有者，数据拥有者是指拥有图数据并且需要离线计算一次隐私保护的2-hop索引的一方。在建立好基于隐私保护的2-hop索引后，数据拥有者将索引数据外包给一个第三方服务提供商。并且，它还对已授权的用户提供一个盐随机因子 $s_2$ 和一个私有密钥 $K$ 。盐随机因子 $s_2$ 是用来对他的查询进行加密操作，私有密钥 $K$ 用来对返回的结果进行解密。
- 服务提供商( $\mathcal{SP}$ )，服务提供商通常具有非常强大的计算能力(例如云计算平台)同时有非常专业的服务维护知识。服务提供商可以代替数据拥有者在密文基础上处理高并发的查询请求。并将计算好的密文结果返回给用户端。

- 客户端，客户端在进行可达性查询 $\text{Reach}(u,v)$ 时，利用从数据拥有者处获得的 $s_2$ 来对两个查询点进行哈希加密，并将加密后的查询请求发送到服务商 $\mathcal{SP}$ ，当客户端从 $\mathcal{SP}$ 处得到的返回的查询结果，用私钥 $K$ 对返回的结果进行解密得到关于可达性查询的最终结果。在这里面我们假设客户端和服务商 $\mathcal{SP}$ 是不可以重叠的，也即 $\mathcal{SP}$ 不能是授权的客户端，否则它可以从数据拥有者获得两个密钥 $s_2$ 和 $K$ ，进而它可以还原整个图，造成了隐私的泄露。

**隐私保护目标** 在本文中，我们的隐私保护目标主要为了保护以下的两个方面的隐私，使得攻击者无法获取以下两方面的信息。

- 查询点之间的可达性信息，这主要是希望对于给定的一个可达性查询 $\text{Reach}(u,v)$ ，我们不希望攻击者能够推断出是否顶点 $u$ 可以到达顶点 $v$ 。
- 图数据结构，我们的目标使得攻击者无法从索引，以及通过不断的查询等方面来获取图的结构信息，让他们无法判断原图中两个点之间的边或者路径是否存在。

**攻击模型** 和其他所有的数据外包的研究工作中的假设一样，我们认为服务商 $\mathcal{SP}$ 是诚实但是具有好奇心的。为了描述的简单性，我们通常将 $\mathcal{SP}$ 视为攻击者。在本文中，我们假设 $\mathcal{SP}$ 可以采取以下的攻击方式来对系统进行攻击。

- 基于密文的攻击，我们假设服务商 $\mathcal{SP}$ 只能访问到加密的图数据索引，他们不能获取关于原始图数据的任何信息。
- 基于集合大小的攻击，我们假设 $\mathcal{SP}$ 尝试根据索引文件的大小和对查询结果的不同大小来进行信息的猜测。

在安全领域，可能还有很多的其他类型的攻击，例如信道攻击、查询路径攻击以及基于频率的攻击等等，为了系统分析的简单性和准确性，我们在本文中，不对此类攻击进行防范和保护。

### 3.4 本章小结

在本章中，我们对原始的2-hop索引方法进行了详细介绍，并对哈希方法和乘法同态加密算法进行了介绍。在本章最后，我们对本文中的研究问题，就数据模型、系统模型、隐私保护目标和攻击模型等四个方面对本文的研究问题进行定义。

## 第四章 图上可达性查询的隐私保护算法

在本章中，我们重点详细介绍隐私保护的2-hop索引算法。在4.1节中，我们介绍一种新的启发式算法来代替maxDensCover，使用该启发式算法能够使得2-hop索引最终的 $I_{\max}$ 能够尽可能的最小化。在4.2中，我们提出了一种新的贪心算法，通过往4.1中得到的2-hop索引的Lins和Louts中加入尽可能少的代理点使得最终的所有Lins和Louts两两的交集大小是完全一致。在4.3中，我们通过一定的加密算法对2-hop索引进行加密。在4.4中，我们详细介绍在加密的索引条件下，安全的可达性查询如何进行。在4.5中，我们对本文提出的隐私保护算法的隐私保护的作用进行分析。最后，我们通过实验对本文中提出的算法效果与效率进行实验的验证。

### 4.1 具有Imax感知的2-hop算法

首先我们给出 $I_{\max}$ 的定义：

**定义 4.1** 对于给定的一个图 $G$ 的2-hop索引， $I_{\max} = \forall u, v, \max(|\text{Lout}(u) \cap \text{Lin}(v)|)$ ，其中 $u, v \in V$ 。

在实际中通过maxDensCover启发式条件构建出的 $I_{\max}$ 通常非常大。在我们的实验中，通过maxDensCover条件建立出来的 $I_{\max}$ 平均为378。在实验中，我们会发现比较大的 $I_{\max}$ 通常会导致需要加入非常多的人工中心点以及会导致比较大的查询代价。

在本节中，我们提出一个新的启发式条件使得在建立2-hop的过程中能够使得 $I_{\max}$ 尽可能的最小。其主要思想是在建立2-hop索引的过程中就将交集的信息考虑进去。详细的来说我们的目标主要是要去最小化以下的两个指标：

1. 和最原始的2-hop工作中一样，我们期望每次选择能够覆盖到未被覆盖的传递闭包集合 $T'$ 中最多的元素的这样一个中心点 $w$ ，通过这样的选择可以带来两方面的好处，一是可以使得中心点的数目较少，二是可以使2-hop的索引大小较小；

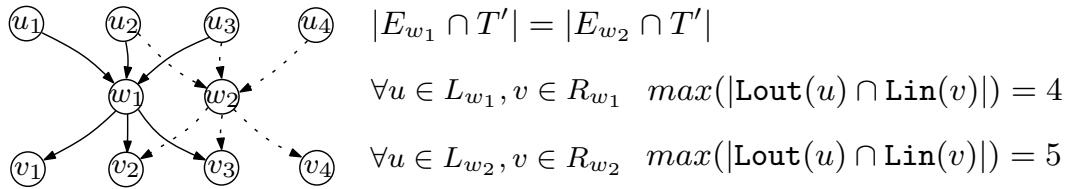


图 4.1: 使用maxISCover 选择中心点的举例

2.  $\max(|\text{Lout}(u) \cap \text{Lin}(v)|)$ , 也即使得所有  $\text{Lout}(u)$  和  $\text{Lin}(v)$  之间交集最大的交集大小能够尽可能的小。

为了实现以上的两个目标, 我们提出了以下的启发式函数, 我们把它称之为maxISCover, 具体的定义如下:

$$\text{maxISCover} = \frac{|E_w \cap T'|}{\max(|\text{Lout}(u) \cap \text{Lin}(v)|)}, \quad (4.1)$$

这里  $u \in L_w, v \in R_w, T'$  是传递闭包集合  $T(G)$  中还未覆盖的的传递闭包集合条目。公式 4.1 主要包括以下两个部分:

1.  $E_w \cap T'$  表示如果我们选择  $w$  为中央点的话, 那么通过选择这个中心点可以覆盖到  $T(G)$  中未覆盖到的元素的条目数;
2.  $\max(|\text{Lout}(u) \cap \text{Lin}(v)|)$  表示对于所有的  $u, v \in V(G)$  中  $|\text{Lout}(u) \cap \text{Lin}(v)|$  最大的交集集合大小。

上面 4.1 将以上的两个目标集合起来表示, 我们希望在选择一个中心点的时候, 即可以包含尽可能多的未被覆盖到的传递闭包集合的元素, 但是同时使得所有的  $\text{Lout}$  和  $\text{Lin}$  之间的交集尽可能的小, 以达到我们希望在建立 2-hop 索引时既可以使得索引的大小较小同时使得  $I_{\max}$  也尽可能的小。所以在我们的工作中我们用 4.1 公式中的 maxISCover 替代 3.1 节中公式 3.1 定义的 maxDensCover。

**例子 1** 我们通过图 4.1 中的一个例子来说明我们的 maxISCover 启发式方法和 [17] 中 maxDensCover 启发式方法的区别。图中左边为图数据, 假设在这一次迭代循环中, 只有两个中心 ( $w_1$  和  $w_2$ ) 可能作为被选择的备选点。从图中可以看出  $L_{w1} = \{u_1, u_2, u_3\}$ ,  $R_{w1} = \{v_1, v_2, v_3\}$ ,  $L_{w2} = \{u_2, u_3, u_4\}$ ,  $R_{w2} = \{v_2, v_3, v_4\}$ 。我们假设  $|E_{w1} \cap T'| = |E_{w2} \cap T'|$ , 那么在 maxDensCover 算法中, 在这一次迭代的循环中, 我们既可以选择  $w_1$  作为中心点, 也可以选择  $w_2$ 。再进一步假设  $\forall u \in L_{w1}, v \in R_{w1}, \max(|\text{Lout}(u) \cap \text{Lin}(v)|) = 4$  以及  $\forall u \in L_{w2}, v \in R_{w2}, \max(|\text{Lout}(u) \cap \text{Lin}(v)|) = 5$ 。那么如果在我们的 maxISCover 算法中, 我们只能选  $w_1$  作为这一次循环的中心点, 因为选择它带来的好处是我们可以覆盖同样数目的未被覆盖的传递闭包集元素, 同时也可以使得全局的 Lin 和 Lout 的交集较小。

## 4.2 代理节点的添加算法

在本节中我们从两个方面来介绍我们的代理节点添加算法。一方面对在上一节中利用maxISCover启发式方法建立出的2-hop索引通过加入一些人工代理中心点使得任意Lout 和Lin 之间的交集大小一致。另一方面，我们通过添加人工代理中心点使得所有的Lout和Lin集合里的元素数目都具有一个用户指定阈值的差异，以防止SP通过基于大小的攻击方法来对索引的信息进行猜测。

### 4.2.1 交集大小归一化

交集大小归一化主要处理的问题是如何通过往2-hop索引(Lins和Louts)中加入一些人工的代理节点来实现对于每一个 $u \in V(G)$ 和 $v \in V(G)$ ，使得添加了代理节点后的Lout<sup>s</sup>(u) 和Lin<sup>s</sup>(v)之间的交集大小为一个定值 $I_{\max}$ 。关于 $I_{\max}$ ，由于在一些非常特殊的图，比如图中所有的点基本都是孤立的，在这些非常离散的图中 $I_{\max}$  要么为0要么为1，那么攻击者可能很容易通过最后的交集大小猜测到这些图的一些结构信息，比如这个图非常的离散。为了使得我们的系统能够更安全，对于这些非常特殊的图，我们可能将 $I_{\max}$  设立一个最小值，比如设置 $I_{\max}=4$ ，这样攻击者SP就无法从最终的 $I_{\max}$ 获取到图的任何信息。关于安全部分的分析，具体可见 4.5。在本节中，我们注重提出我们的交集大小归一化算法，使得通过加入最少的人工代理点让所有集合的交集大小为 $I_{\max}$ 。由于对任何的交集大小都是一样的，那么攻击者SP就不能从交集的结果得到关于两个查询点之间的可达性信息(在2-hop索引中，对于一个查询Reach(u,v)，Lout(u) ∩ lin(v)的集合大小表明了顶点之间的可达性信息，只有不可达的点之间的交集大小为0)。我们将这个问题称之为最少代理中心点添加(*minimum addition of surrogate nodes, MASN*)问题。我们下面给出MASN问题的定义。

**定义 4.2** 最少代理中心点添加问题(MASN)是对于给定一个图G的2-hop索引，通过向Lins 和Louts 中添加一些人工的代理节点得到Lin<sup>s</sup>s 和Lout<sup>s</sup>s使得：

- $\forall u, v \in V(G), |\text{Lout}^s(u) \cap \text{Lin}^s(v)| = I_{\max}$ ;
- Lin<sup>s</sup>和Lout<sup>s</sup>集合中最大的集合元素都最小化;
- $\sum_{u \in V(G)} (|\text{Lin}^s(u)| + |\text{Lout}^s(u)|)$ 也最小化。

**引理 4.1** 最少节点添加问题(MASN)是一个NP-hard问题。

我们给出对于引理 4.1的证明，最少代理中心点添加问题的复杂度可以从一个经典的最小顶点覆盖问题得到。关于最小顶点覆盖问题(MINIMUM VERTEX

COVER problem (MVC))定义如下：给定一个图 $G = (V, E)$ ，寻找顶点集 $V$ 的一个最小子集 $V'$ 使得对于图中的每一条边 $(u, v) \in E$ ，顶点 $u$ 或者顶点 $v$ 至少有一个在集合 $V'$ 中。

考虑最小顶点覆盖问题的一个实例图 $G = (V, E)$ ，我们可以通过以下的方法构造一个最少代理节点添加问题的实例：对于任何一个 $v_i \in V$ ， $v_i$ 和一个可以添加到2-hop的Lins或Louts中的一个代理节点 $d_i$ 相对应，图中的一条边 $(v_i, v_j) \in E$ 表示集合 $\text{Lin}(a_i) \cap \text{Lout}(b_j)$ 的一个交集。我们通过特殊的构造方法构造出Lins和Louts使得其满足以下两个条件：

1. 通过添加 $d_i$ 或 $d_j$ 到 $\text{Lin}(a_i)$ 和 $\text{Lout}(b_j)$ 中，使得 $|\text{Lin}(a_i) \cap \text{Lout}(b_j)|$ 恰好等于 $I_{\max}$ ；
2. 添加 $d_i$ 或 $d_j$ 到 $\text{Lin}(a_l)$ 或 $\text{Lout}(b_m)$ 中，在图 $G$ 表示为一条边 $(v_l, v_m)$ ，将使得 $|\text{Lin}(a_l) \cap \text{Lout}(b_j)| > I_{\max}$ 或者 $|\text{Lin}(a_i) \cap \text{Lout}(b_m)| > I_{\max}$ ，这里 $v_i \neq v_l$ ， $v_j \neq v_m$ 。

---

**Algorithm 1** 交集大小归一化算法unifyIS(Lout, Lin)

---

**Input:** 2-hop索引Lins 和Louts

**Output:** Lout<sup>s</sup>s和Lin<sup>s</sup>s

- 1: 计算 $I_{\max}$  大小
  - 2: 初始化代理节点集合 $D_w = []$
  - 3: 对图中的顶点 $V(G)$ 创建一个优先级队列 $Q_v$ ，在队列中，各个点 $v \in V(G)$ 的优先级定义如下：  

$$\max_{u \in V} (I_{\max} - |\text{Lout}(u) \cap \text{Lin}(v)|)$$
  - 4: **while**  $v \neq \text{null}$ , 其中 $v \leftarrow Q_v.\text{getNext}()$  //扫描Lins
  - 5:    $D_w.\text{movetofront}()$  //将数组指针移到 $D_w$ 的最前面
  - 6:   **while**  $d_i \leftarrow D_w.\text{getNext}()$
  - 7:     **if**  $d_i = \text{null}$  并且 $\exists u', v' | \text{Lout}(u') \cap \text{Lin}(v')| < I_{\max}$
  - 8:        $d_i \leftarrow \text{new node}()$  同时 $D_w.\text{push}(d_i)$
  - 9:     **else** break
  - 10:
  - 11:    **for each**  $u \in V(G)$ , 且 $|\text{Lout}(u) \cap \text{Lin}(v)| < I_{\max}$  //扫描Louts
  - 12:      //如果交集大小约束条件满足
  - 13:       **if**  $\psi(2\text{-hop}, u, v, d_i)$  条件为真
  - 14:           $\text{Lout}(u) \leftarrow \text{Lout}(u) \cup \{d_i\}; \text{Lin}(v) \leftarrow \text{Lin}(v) \cup \{d_i\}$
  - 15: **return** Louts和Lins中添加了代理中心点后的Lout<sup>s</sup>s和Lin<sup>s</sup>s
- 

以上的两条规则说明了 $d_i$  或 $d_j$ 只能被加入到 $\text{Lin}(a_i)$  和 $\text{Lout}(b_j)$ 中。最后我们再将原先真实的中心点加入到Lin和Lout中使得得到的Lins和Louts都分

别具有相同的大小。因此MASN问题关于使得 $\text{Lin}_{\max}$ 和 $\text{Lout}_{\max}$ 最小化的限制在这里变得无效。通过这种方法得到的2-hop索引的大小最大为 $|V|I_{\max} \times 2|V|$ , 这里 $\text{maxintr}$ 是对于每一个 $v_i \in V$ 需要满足第二个条件要添加的点的数量,  $2|V|$ 是所有的 $\text{Lins}$ 和 $\text{Louts}$ 的数量。

假设我们已经找到关于MASN中代理中心点集合 $D_w$ 的一个解决方案, 在该算法中所有的交集大小都是 $I_{\max}$ 。那么首先,  $d_i$ 或者 $d_j$ 已经被添加到所有的可能的 $\text{Lin}(a_i) \cap \text{Lout}(b_j)$ 的集合中。这里每一个可能的集合都是一条边, 表示为 $C = \{v_i - d_i \in D_w\}$ , 那么 $C$ 就是一个最小顶点覆盖。我们可以很容易的得到只有在 $C$ 能够有最小化的答案时,  $D_w$ 才能够有最小化的答案。因此我们从MASN的一个解决方案中得到了MVC的一个解决方法, 但是由于MVC是一个NP-hard问题, 所以以上的MASN也是一个NP-hard问题。

针对以上的MASN这样一个NP-hard问题, 在本文中, 我们提出了一个贪心算法, 叫做unifyIS算法来解决这一问题。关于unifyIS算法详细见算法1。unifyIS算法的输入是前面一章节中算法生成的2-hop索引, 也就是 $\text{Lins}$ 和 $\text{Louts}$ 。算法的输出为添加了代理中心点的2-hop索引, 为了与前面的标识相区别, 我们将其标记为 $\text{Lout}^s$ s和 $\text{Lin}^s$ s。算法的主要思想是对于每一个代理中心点 $d$ 我们期望它可能加入尽可能多的 $\text{Lins}$ 和 $\text{Louts}$ 中, 规模为 $O(|V|)$ 。那么通过往 $O(|V|)$ 规模的 $\text{Lins}$ 和 $\text{Louts}$ 中加入一个代理中心点使得 $T(G)$ 中有 $O(|V|^2)$ 级别的元素能够受益于当前加入的代理中心点。具体来说, 在算法的第2行中, 我们使用了一个队列 $D_w$ 来保存当前已加入到2-hop索引中的代理中心点的ID。在算法第3-4行, unifyIS首先选择处理 $\text{Lins}$ 与其他的 $\text{Louts}$ 的交集与 $I_{\max}$ 有较大差距的 $\text{Lins}$ 。因为之前已经加入到部分的 $\text{Lins}$ 中的代理中心点可以用来减小当前的 $\text{Lins}$ 与其他 $\text{Louts}$ 之间的差距, 所以在算法的第5行中, 对代理中心点队列 $D_w$ 进行扫描, 寻找当前的代理中心点集合中可以重复利用的中心点。在算法的第6-14行, 算法对于每一个 $\text{Louts}$ 进行处理。在第7行中, 算法检查是否存在一个 $\text{Lout}$ 需要被处理, 也即 $\text{Lin}$ 与当前的 $\text{Lout}$ 的交集大小小于 $I_{\max}(|\text{Lout}(u) \cap \text{Lin}(v)| < I_{\max})$ 。如果以上的条件满足, 那么如果之前的所有代理节点已经被利用, 那么就新生成一个代理节点, 否则如果上面的条件不满足, 则算法跳出这次迭代。如果对于选中的一个代理节点, 我们检验如果将其加入到 $\text{Lout}(u)$ 和 $\text{Lin}(v)$ 中如果能否满足:

$$\begin{aligned} \psi(2\text{-hop}, u, v, d_i) : \quad & \forall v', |(\text{Lout}(u) \cup \{d_i\}) \cap \text{Lin}(v')| \leq I_{\max} \wedge \\ & \forall u', |\text{Lout}(u') \cap (\text{Lin}(v) \cup \{d_i\})| \leq I_{\max}, \end{aligned} \tag{4.2}$$

其中 $u', v' \in V(G)$ 。

如果当前的代理中心点 $d_i$ 能够满足上面的条件 $\psi$ , 那么我们则将其添加到 $\text{Lout}(u)$ 和 $\text{Lin}(v)$ 。该算法当所有的 $u, v \in V(G)$ 满足 $|\text{Lout}(u) \cap \text{Lin}(v)| =$

$v$	$\text{Lin}^s(v)$	$\text{Lout}^s(v)$
0	<b>0, 7, 8, 9</b>	<b>0, 1, 2, 3, 5, 6, 7, 8</b>
1	<b>1, 7, 8, 10</b>	<b>1, 2, 3, 5, 6, 7, 8, 9</b>
2	<b>2, 7, 8, 11</b>	<b>2, 3, 5, 6, 7, 8, 9, 10</b>
3	<b>3, 7, 8, 12</b>	<b>3, 4, 7, 8, 9, 10, 11</b>
4	<b>0, 1, 2, 4, 9, 10, 13</b>	<b>4, 7, 8, 9, 10, 11, 12</b>
5	<b>4, 5, 7, 8, 14</b>	<b>5, 6, 7, 8, 9, 10, 11, 12, 13</b>
6	<b>4, 6, 7, 8</b>	<b>6, 7, 8, 9, 10, 11, 12, 13, 14</b>

图 4.2: 添加完代理节点后的2-hop索引

$I_{\max}$ 时，算法结束。

算法unifyIS在最坏的情况下需要往2-hop索引中加入 $I_{\max}|V|$ 个不同的代理中心点，在最好的情况下只需要 $I_{\max}$ 个代理中心点。因此，unifyIS算法在最坏的情况下，需要向2-hop中加入总共 $I_{\max}|V|(1 + |V|)$ 个点，那么也即使索引的大小增加了 $I_{\max}|V|(1 + |V|)$ ，在最好的情况下，我们总共只会使原来的2-hop索引大小增加 $2 \cdot I_{\max}|V|$ ，这里 $|V|$ 是图 $G$ 中顶点的个数。

**例子 2** 在图 4.2中我们展示了在图 3.1中的原始2-hop索引的基础上，通过向索引里加入代理中心点使得任意两个集合的交集大小变为一致。在图中，我们使用加粗的ID表示是图数据原始的2-hop索引中真实的中心点，其他非粗体的ID表示我们人工加入的代理中心点。在该图中，它的 $I_{\max}=3$ 。

我们首先来考虑算法unifyIS的第一次迭代，由于 $I_{\max} - |\text{Lin}(0)| \cap |\text{Lout}(1)| = I_{\max}$ ，所以顶点0是当前优先级队列中排名最高的一个顶点，同时很明显，很多的Lin和Lout之间的交集大小明显小于 $I_{\max}$ ，并且此时由于 $D_w$ 为空集，所以暂时没有代理节点可以利用，故我们在第8行程序中新建了一个代理节点7。由于代理节点7并不违反 $\psi$ 条件，所以我们将代理节点7加入到集合 $\text{Lin}(0)$ 和 $\text{Lout}(1)$ 中，使得这两个集合之间的交集大小增加1，同时由于 $\text{Lin}(0)$ 和 $\text{Lout}(0)$ ， $\text{Lout}(2-6)$ 之间的关系都不违反 $\psi$ 条件，所以我们将代理节点7添加到集合 $\text{Lout}(0)$ ， $\text{Lout}(2-6)$ 中。在接下来的几次循环中，unifyIS添加了新的代理节点8和9到 $\text{Lin}(0)$ 和相应的 $\text{Louts}$ 中使得 $\text{Lin}(0)$ 和所有的 $\text{Louts}$ 的交集大小都均为 $I_{\max}=3$ 。

在节点0的Lin处理完之后，unifyIS选择优先级队列中下一个顶点，比如1，然后unifyIS算法首先在已经存在的代理节点中查找看是否有可以重复利用的代理节点，比如代理节点7，因为在 $\text{Lin}(1)$ 中加入该代理节点不会违反 $\psi$ 条件，所以将代理节点7加入到 $\text{Lin}(1)$ ，是的 $\text{Lin}(1)$ 与含有代理节点7的 $\text{Louts}$ 之间的交集大小增加1。基于同样的原理，代理节点8也被加入到 $\text{Lin}(1)$ 中。但是，代理节点9不能再被重复利用，因为此时 $|\text{Lin}(1) \cap \text{Lout}(1)| = |\{1, 7, 8\}| = I_{\max}$ ，如果我们再将9加入 $\text{Lin}(1)$ ，那么会使得 $\text{Lin}(1)$ 和 $\text{Lout}(1)$ 的交集大小变为4，从而大于了 $I_{\max}$ ，违反了条件的要求。

我们考虑图 4.2 中的代理节点 7，它被插入到 13 个不同的集合中，然后它却覆盖到  $T(G)$  中的  $6 * 7$  个元素，我们通过重复利用代理节点，使得 2-hop 索引增加的大小最小化。通过往 2-hop 索引中添加入一些人工的代理节点，我们使得任意的查询点  $u$  和  $v$ ，他们之间的交集大小始终为定值  $I_{\max}$ 。我们仍然考虑前面例子中的查询， $\text{Lout}^s(1) \cap \text{Lin}^s(5) = \{5, 7, 8\}$  和  $\text{Lout}^s(6) \cap \text{Lin}^s(0) = \{7, 8, 9\}$ ，对于原先无论可达或不可达的查询，他们的结果始终是一样的大小，这样也就使得攻击者无法根据查询结果判断两个查询点之间的可达性关系，因为任何的查询，其查询结果交集大小是完全一样的。

#### 4.2.2 Lin 和 Lout 大小归一化

由于在上一节中，最少代理中心点添加问题已经是 NP-hard 问题了，所以我们在算法中只考虑了如何使得任何两个集合之间的交集大小变为一致，但是通常由 unifyIS 算法建立出来的  $\text{Lin}^s$ s 和  $\text{Lout}^s$ s 集合的元素个数差距会比较大。为了防止攻击者通过利用集合大小的信息对图中一些中心点的链接关系进行猜测，我们在这一节提出了一种后处理的  $\text{Lin}^s$ s 和  $\text{Lout}^s$ s 集合大小归一化的处理算法 unifyLin。由于对  $\text{Lin}^s$  和  $\text{Lout}^s$  的归一化是完全一样的一个过程，所以为了文章的简洁性，我们在这里只讨论如何归一化  $\text{Lin}^s$ ，对于  $\text{Lout}^s$  可以使用和  $\text{Lin}^s$  完全一样的算法进行处理。

我们使用  $\text{Lin}_{\max}$  和  $\text{Lout}_{\max}$  分别表示在  $\text{Lin}^s$ s 和  $\text{Lout}^s$ s 中具有最大集合大小的集合  $\text{Lin}^s$  和  $\text{Lout}^s$ 。我们的算法的主要思想基于以下的两方面想法：(1) 对于图  $G$  中的任何一个顶点  $u \in V(G)$ ，如果  $|\text{Lin}^s(u)| < \text{Lin}_{\max}$ ，我们通过将  $\text{Lin}^s(u)$  集合里面的代理节点进行分裂，使得  $\text{Lin}^s(u)$  的大小接近  $\text{Lin}_{\max}$  的大小；(2) 我们在对任何点的分裂过程中，算法要保证不增加  $I_{\max}$  的大小。下面我们给出这个问题的一个严格定义，我们称该问题为索引大小归一化问题 (Unification of the Labeling Size, ULS)。

**定义 4.3** 索引大小归一化问题 (*Unification of the Labeling Size, ULS*) 是指给定集合  $\text{Lin}^s$ s 和  $\text{Lout}^s$ s，我们希望通过一定的算法使得它们的集合大小能够  $\forall u, v$ ,

$$\frac{|\text{Lin}^s(u)| - \text{Lin}_{\max}}{\text{Lin}_{\max}} \leq \delta \text{ 和 } \frac{|\text{Lout}^s(u)| - \text{Lout}_{\max}}{\text{Lout}_{\max}} \leq \delta,$$

在这里  $\delta$  是一个用户指定的参数，该参数规定索引大小之间的差距允许的误差。

我们在算法 2 中介绍了 unifyLin 算法来解决索引归一化问题。假设我们首先对所有的  $\text{Lin}^s$  按照它们集合的大小进行从大到小进行排序，我们根据排序的情况，对所有的  $\text{Lin}^s$  使用 unifyLin 算法。我们首先在算法的第 1-2 行中，从集合  $\text{Lin}^s(u)$  中选择一个人工代理中心点  $w$ 。在算法的第 3 行中，我们根据选择的代

**Algorithm 2**  $\text{Lin}^s$ s集合大小归一化算法  $\text{unifyLin}(\text{Lout}^s, \text{Lin}^s, \delta)$ 

**Input:** 添加完了代理节点后的2-hop索引  $\text{Lout}^s$  和  $\text{Lin}^s$ , 以及用户自定义的误差阈值的参数  $\delta$

**Output:**  $\text{Lout}^s$  和  $\text{Lin}^s$

- 1: **for each**  $u \in V(G)$ ,  $|\text{Lin}^s(u)| + \text{Lin}_{\max} * \delta \leq \text{Lin}_{\max}$
  - 2:   从集合  $\text{Lin}^s(u)$  中选取一个人工代理节点  $w$
  - 3:   将原先的代理节点  $w$  分裂成一个集合  $W = \{w_1, \dots, w_n, w_{n+1}\}$ ,  $n \leq \text{Lin}_{\max} - |\text{Lin}^s(u)| - 1$
  - 4:   我们将点  $w$  从原先的集合  $\text{Lin}^s(u)$  中移除并将集合  $W$  中的所有元素添加到集合中, 即  $\text{Lin}^s(u) \leftarrow (\text{Lin}^s(u)/\{w\}) \cup \{w_1, \dots, w_{n+1}\}$
  - 5:   **for each**  $u' \in V(G)$ ,  $|\text{Lin}^s(u')| < \text{Lin}_{\max} \wedge w \in \text{Lin}^s(u')$
  - 6:      $\text{Lin}^s(u') \leftarrow (\text{Lin}^s(u')/\{w\}) \cup \{w_{n+1}, w_{n+2}\}$
  - 7:   **for each**  $v \in V(G)$ ,  $|\text{Lout}^s(v)| = \text{Lout}_{\max} \wedge w \in \text{Lout}^s(v)$
  - 8:      $\text{Lout}^s(v) \leftarrow \text{Lout}^s(v) \cup \{w_{n+1}\}$
  - 9:   **for each**  $v \in V(G)$ ,  $|\text{Lout}^s(v)| < \text{Lout}_{\max} \wedge w \in \text{Lout}^s(v)$
  - 10:     $\text{Lout}^s(v) \leftarrow \text{Lout}^s(v) \cup \{w_i, w_{n+2}\}$
- 其中  $i \in [1, n]$ , 并且每一个  $i$  都至少一次被加入到  $\text{Lout}^s$  中

$v$	$\text{Lin}^s(v)$	$\text{Lout}^s(v)$
$\vdots$	$\vdots$	$\vdots$
4	<b>0, 1, 2, 4, 9, 10, 13</b>	<b>4, 7, 7<sub>1</sub>, 7<sub>3</sub>, , 8, 9, 10, 11, 12</b>
5	<b>4, 5, 7<sub>1</sub>, 7<sub>2</sub>, 8, 14</b>	<b>5, 6, 7, 7<sub>2</sub>, 8, 9, 10, 11, 12, 13</b>
6	<b>4, 6, 7<sub>2</sub>, 7<sub>3</sub>, 8</b>	<b>6, 7, 7<sub>2</sub>, 8, 9, 10, 11, 12, 13, 14</b>

Encryption of  $\text{Lin}^s(6)$ :

$$\text{Lin}^e(h_{s_2}(6)) = \{(h_{s_1}(4), E(0)), \dots, (h_{s_1}(8), E(1))\}$$

图 4.3:  $\text{unifyLin}$  算法一次迭代的中间结果以及  $\text{Lin}^s(6)$  的加密

理中心点生成一个新的代理中心点集合  $\{w_1, \dots, w_n, w_{n+1}\}$ , 其中  $n \leq \text{Lin}_{\max} - |\text{Lin}^s(u)| - 1$ 。在算法第4行中, 我们使用  $\{w_1, \dots, w_{n+1}\}$  代替原先  $\text{Lin}^s(u)$  中的节点  $w$ 。在算法的第5-6行, 我们对于任何一个  $u' \in V(G)$ ,  $w \in \text{Lin}^s(u')$  且  $|\text{Lin}^s(u')| < \text{Lin}_{\max}$ , 我们使用  $w_{n+1}$  和  $w_{n+2}$  代替集合  $\text{Lin}^s(u')$  中原先的  $w$ 。在算法的第7-8行, 对于那些  $v \in V(G)$ , 它的  $|\text{Lout}^s(v)| = \text{Lout}_{\max}$  并且  $w \in \text{Lout}^s(v)$ , 我们则向这些  $\text{Lout}^s(v)$  集合中添加一个新的代理节点  $w_{n+1}$ 。在算法的第9-10行, 对于所有的满足  $|\text{Lout}^s(v)| < \text{Lout}_{\max}$  且  $w \in \text{Lout}^s(v)$  的顶点  $v$ , 我们在集合  $\text{Lout}^s(v)$  中添加新的代理节点  $\{w_i, w_{n+2}\}$ , 其中  $i \in [1, n]$ 。

**例子 3** 我们利用图 4.2 中的已经使用算法 1 处理完的 2-hop 索引来说明算法 2。利用算法  $\text{unifyLin}$  处理完的索引见图 4.3。假设我们从  $\text{Lin}^s(5)$  中选择代理中心点 7 作为分裂中心点, 由于  $\text{Lin}_{\max} = 7$  并且  $|\text{Lin}^s(5)| = 5$ , 所以  $n + 1$  可以是 2, 我

们将代理中心点7分裂为 $\{7_1, 7_2\}$ 。我们用 $\{7_1, 7_2\}$ 替换集合 $\text{Lin}^s(5)$ 中的代理中心点7。同时，由于代理中心点7也出现在一些其他的 $\text{Lin}^s$ 中，我们对于这些集合，我们向其中添加新的代理中心点 $\{7_2, 7_3\}$ 到集合 $\text{Lin}^s(i), i \in \{0, 1, 2, 3, 6\}$  中。由于 $\text{Lout}_{\max} = |\text{Lout}^s(5)| = |\text{Lout}^s(6)|$ ，我们将新的代理中心点 $7_2$ 添加到 $\text{Lout}^s(5)$ 和 $\text{Lout}^s(6)$ 中。对于 $\text{Lout}^s(j), j = \{0, 1, 2, 3, 4\}$ ，我们向其中添加 $7_3$ 。

我们可以使用一个很简单的例子来分析，经过unifyLin算法处理过以后的2-hop索引的任意集合之间的交集大小并没有被改变。与此同时，由于使用unifyLin算法而产生的一些新的代理节点将出现在部分交集结果中。通过使用unifyLin算法，2-hop索引 $\text{Lin}^s$ 和 $\text{Lout}^s$ 增加的大小主要有以下几部分：(1) $\text{Lin}^s(u)$ 集合增加的大小为 $n + 1$ ；(2)那些含有选中的代理中心点 $w$ 的 $\text{Lin}^s$ 增加的大小为2；(3)对于含有代理中心点 $w$ 的 $\text{Lout}^s$ ，如果它的集合大小已经等于了 $\text{Lout}_{\max}$ ，那么它增加的大小为1；(4)对于含有代理中心点 $w$ 的 $\text{Lout}^s$ ，如果它的集合大小小于 $\text{Lout}_{\max}$ ，那么它增加的大小为2。所以在unifyLin算法中，我们总使得距离目标远的集合的增长速度大于那些距离目标近的集合。然后我们可以交替的在 $\text{Lin}^s$ 和 $\text{Lout}^s$ 上使用unifyLin算法直到每一类集合之间大小的差距满足我们在定义 4.3 中定义的一个阈值 $\delta$ 。

### 4.3 索引加密处理算法

在4.2中我们已经介绍了如何使用unifyIS算法和unifyLin算法来对原始的2-hop索引进行添加人工代理中心点使得任意两个集合之间的交集大小都等于 $I_{\max}$ ，同时使得所有的 $\text{Lin}^s$ 和 $\text{Lout}^s$ 的集合大小浮动都能在一个规定的阈值范围内。经过以上的处理，在本节中，我们着重介绍如何去对这些索引进行加密处理。

为了区分在索引中，哪一些中心点是真实的中心点，哪一些中心点是我们通过算法添加的人工代理中心点，我们对于2-hop索引中的每一个中心点使用一个标志来标识它的真实与否，具体见定义 4.4。对于是真实中心点的，该标志位为0，否则该标志位为1。我们将在 4.4 中介绍如何使用该标志位来对查询结果进行加密。下面给出在 $\text{Lin}^s$ s 和 $\text{Lout}^s$ s 中新的中心点的定义。

**定义 4.4** 2-hop索引( $\text{Lout}^s(u)$ 和 $\text{Lin}^s(v)$ )中的每一个中心点都是一个二元组( $w, f$ )，当该 $w$ 中心点是一个原始2-hop中真实的中心点时 $f = 0$ ，否则该标志位为1。

基于定义 4.4，我们对所有的中心点进行加密使得既可以保护查询点之间的可达性信息也可以保护图的结构信息。(1)为了隐藏图中顶点和索引中

中心点的任何联系，我们对 $(w, f)$ 中的 $w$ 和 $\text{Lout}^s(u)$ ,  $\text{Lin}^s(u)$ 中的 $u$ 使用带有不同盐参数的单向防冲突哈希函数来对这两类不同的点进行映射，我们把这两个不同的哈希函数分别标记为 $h_{s_1}(w)$  和 $h_{s_2}(u)$ 。对于可能 $w = u$ , 但是并不表示 $h_{s_1}(w) = h_{s_2}(u)$ , 其中 $s_1 \neq s_2$ 。(2)关于二元组中标志位的加密，我们使用乘法同态加密算法 $E(\cdot)$  [19]来对其进行加密。采用Elgamal加密算法主要有以下两点好处：(1)由于标志位只有两种不同的值0或者1, Elgamal加密算法在加密时引入了随机数使得相同的值可以被加密成不同的密文，所以使用Elgamal加密算法能够使得标志位加密后的值变得多样化，也使得攻击者无法通过对相同的标志位进行统计来猜测到一些信息；(2)由于Elgamal加密算法是乘法同态的加密算法，所以它可以允许客户端只用进行一次解密就能够得到查询结果，这一部分详细见 4.4介绍。至此，我们已经完成了完整的2-hop索引的创建，所以在此我们给出隐私保护的2-hop(privacy-preserving 2-hop , pp-2-hop) 索引的精确定义。

**定义 4.5** 对于一个图 $G = (V(G), E(G))$ , 它的隐私保护的2-hop(pp-2-hop)索引就是对于图中每个加密的顶点 $u_e$ ,  $u_e = h_{s_2}(u)$  都带有两个加密的集合 $\text{Lout}^s(u)$ 和 $\text{Lin}^s(u)$ (记着 $\text{Lout}^e(u)$ ,  $\text{Lin}^e(u)$ )表示顶点 $u$ 的可达性信息。其中对于集合 $\text{Lout}^e(u)$ ,  $\text{Lin}^e(u)$ 中的每一个元素都是一个二元组 $(w_e, f_e)$ , 其中 $w_e = h_{s_1}(w)$  ,  $f_e = E(f)$ 。

**例子 4** 在图 4.3中，我们就节点6的 $\text{Lin}^e(6)$ 作为一个例子进行展示。对 $\text{Lin}^e(6)$ 进行加密后的 $\text{Lin}$ 我们记作为 $\text{Lin}^e(h_{s_2}(6))$ , 其中 $h_{s_2}(6)$ 是对节点6进行加密后的密文串。同时，对于每一个集合中的元素，我们也进行加密，例如对于 $\text{Lin}^s(4)$ 集合中的第一个元素 $(4, 0)$ ，我们通过加密使其变为 $(h_{s_1}(4), E(0))$ 。

## 4.4 隐私保护的可达性查询处理

基于在前一节中定义 4.5中定义的加密的pp-2-hop索引，我们在本节中将介绍如何在不解密的条件下进行可达性查询的处理。查询处理主要有以下三个步骤：(1)首先，客户端对查询内容 $\text{Reach}(u, v)$ 进行加密，我们将查询内容从 $\text{Reach}(u, v)$ 通过哈希方法进行加密使其变为 $\text{Reach}(u_e, v_e)$ ，并将其提交到服务商 $\mathcal{SP}$ 进行查询；(2)在服务端 $\mathcal{SP}$ 对集合 $\text{Lout}^e(u_e)$ 和 $\text{Lin}^e(v_e)$ 进行求交集操作，同时将交集后的结果 $R_e$ 返回给客户端；(3)客户端使用从数据拥有者处通过授权得到的密钥 $K$ 通过Elgamal解密算法对返回的结果进行解密并得到最终的结果。

**朴素查询算法** 在以上的三个步骤的前提下，朴素的可达性查询算法是对于一个查询 $\text{Reach}(u, v)$ ，我们在服务端 $\mathcal{SP}$ 做两个集合 $\text{Lout}^e(u_e)$ 和 $\text{Lin}^e(v_e)$ 之间的交集，并将最后交集的结果中的中心点的加密的标志位信息全部返回给客户端。客户端对返回的所有标志位进行一一的解密，检查其中是否至少有一个标志位

为0，表明交集的结果中有至少一个真实的中心点。如果有至少一个标志位解密后为0，则说明两个查询点之间是可达的，否则如果所有的标志位解密后都为1，那么说明这两个点不可达。很明显，由于在前面的算法中，我们使得对于每一个查询，其交集的结果中都有 $I_{\max}$ 个元素，所以在朴素的查询算法中，客户端需要解密 $I_{\max}$ 次，这无疑是非常耗时的。下面我们介绍一种基于乘法同态加密算法的查询处理算法。

**基于乘法同态加密算法的查询算法** 通常我们知道解密是一个比较耗时的过程，尤其是在这种客户端可能具有较弱的计算能力的时候，所以如果能使客户端解密一次那么整个查询时间就会被大大缩短。所以我们提出了一种只需要客户端进行一次解密的查询算法。我们将点 $u_e$ 和 $v_e$ 的 $\text{Lout}^e(u_e)$ 和 $\text{Lin}^e(v_e)$ 交集的结果记为 $R(u_e, v_e)$ 或简记为 $R$ ，那么

$$R = \{(w_e, f_e) \mid (w_e, f_e) \in \text{Lout}^e(u_e) \text{ and } (w_e, f'_e) \in \text{Lin}^e(v_e)\}.$$

那么在基于乘法同态加密算法的查询算法中，由于我们使用的Elgamal算法支持乘法同态特性，也即在密文状态下的乘法运算等于明文状态下的先乘法再进行加密，所以我们将最终的查询结果 $R_e$ 定义为 $\prod_{(w_e, f_e) \in R} f_e$ 。我们将最终结果 $R_e$ 返回给客户端，客户端使用私有密钥 $K$ 对查询结果 $R_e$ 进行一次解密，如果解密完结果为0，则表示两个查询点 $u$ 可以到达 $v$ ，否则如果非0，则表示顶点 $u$ 不可以到达顶点 $v$ 。这里 $R_e$ 是交集结果中很多个标志位乘积的结果，这个乘积为0当且仅当在交集结果中至少包含有一个真实的中心点；如果交集的结果中，所有的点都是我们前面通过算法添加的代理中心点，那么解密后的 $R_e$ 应该为1。

**例子 5** 我们考虑图 3.1 中的图数据，我们考虑在其上进行隐私保护的可达性查询 $\text{Reach}(1, 5)$ 。那么在pp-2-hop索引上的可达性查询算法主要进行以下几个步骤：(1)首先客户端使用从数据拥有者处通过授权获得的盐参数 $s_2$ 对查询点进行哈希使其成为 $h_{s_2}(1)$  和  $h_{s_2}(5)$ ，并将哈希后的两个查询点提交到服务商 $\mathcal{SP}$ ；(2)服务商进行集合交集 $\text{Lout}^e(h_{s_2}(1)) \cap \text{Lin}^e(h_{s_2}(5))$ ，并得到集合交集后的结果 $\{(h_{s_1}(5), E(0)), (h_{s_1}(7), E(1)), (h_{s_1}(8), E(1))\}$ 。基于以上交集得到的结果，服务商进行一个密文状态下的乘法运算得到 $R_e = E(0) \times E(1) \times E(1) = E(0)$ ，并将这个结果返回给查询客户端；(3)客户端使用从图数据拥有者处通过授权获得的解密密钥 $K$ 和 $Elgamal$ 解密程序对返回的 $R_e$ 进行一次解密，发现解密的结果为0，则表明顶点1可以到达顶点5。

## 4.5 图上可达性查询的隐私保护算法的隐私分析

在本节中，我们针对前面在 3.3 节中定义的攻击模型，分析本文的隐私保护算法的安全性。

**针对密文索引的攻击** 我们在下文中证明如果在基于密文的攻击下，本文中的隐私保护可达性查询系统可以保护查询点之间的可达性信息以及图的结构信息不被攻击者  $\mathcal{SP}$  攻击。

**引理 4.2** 攻击者  $\mathcal{SP}$  能够破解查询顶点之间的可达性信息当且仅当  $\mathcal{SP}$  能够破解单向防冲突哈希算法或者 Elgamal 加密算法。

下面我们给出对于引理 4.2 的证明。首先我们分情况讨论一下，如果在攻击者  $\mathcal{SP}$  能够破解哈希算法和 Elgamal 加密算法的情况下，它可以获得的信息，以及在它不能破解以上的两个算法条件下，它能够从系统中获取到的信息。

**情况一：** (1) 我们假设攻击者  $\mathcal{SP}$  可以破解 Elgamal 加密算法，那么  $\mathcal{SP}$  就可以根据中心点的标志位推断出一个中心点是否为真实的中心点。所以在可达性查询期间， $\mathcal{SP}$  可以分析交集结果  $R$ ，那么  $\mathcal{SP}$  就可以根据分析出的交集结果中是否存在一个真实的中心点来判断两个查询点之间的可达性信息。

(2) 如果攻击者  $\mathcal{SP}$  可以破解哈希函数(例如 SHA-1)，那么它就可以破解出  $\text{Lin}^s$  和  $\text{Lout}^s$  中每一个中心点 ID 的未哈希情况下的原文。然后攻击者就可以通过和 pp-2-hop 中的  $\text{Lin}^s$ ,  $\text{Lout}^s$  对比是否有对应的 ID 来判断一个点是否为真实的中心点，进而可以得到查询点之间的可达性信息。

**情况二：** 我们假设攻击者  $\mathcal{SP}$  不能够破解单向防冲突哈希函数和 Elgamal 加密算法。下面我们通过分析在每一步中， $\mathcal{SP}$  可以获取到的信息。对于一个给定的查询  $u_e$  和  $v_e$ ，那么  $\mathcal{SP}$  首先会去 pp-2-hop 索引中检索到  $\text{Lout}^s(u_e)$  和  $\text{Lin}^s(v_e)$ ，然后  $\mathcal{SP}$  求得两个集合的交集，并在基于 Elgamal 加密后的标志位上进行乘法运算进而得到最终的返回结果  $R_e$ 。

首先，由于攻击者  $\mathcal{SP}$  无法破解哈希方法，那么它就不能解密出查询点  $u_e$  和  $v_e$  的真实点的信息，同时，他也不能破解  $\text{Lout}^s(u_e)$  和  $\text{Lin}^s(v_e)$  中中心点的信息，所以它无法在这两类点之间做一个一一映射，也即它不能得到关于查询点的任何信息，对于所有的查询，它每次获取的信息对它而言是完全一样的。同时，由于我们假设攻击者  $\mathcal{SP}$  不能破解 Elgamal 加密算法，那么它也就不能破解  $\text{Lout}^s(u_e)$  和  $\text{Lin}^s(v_e)$  中中心点的标志位信息，进而无法确定一个中心点是真实的中心点或者是人工代理中心点。同时，由于 Elgamal 加密算法支持乘法同态，所以  $\mathcal{SP}$  无法获知  $R_e$  的原文信息，进而  $\mathcal{SP}$  不能获取查询点之间的可达性信息。

在以上，我们利用在进行任何两点的可达性查询时，保护了它们的可达性信息，我们可以证明pp-2-hop可以保护 $\mathcal{SP}$ 不能推测出图的任何结构信息。由于对于任何两点的查询，我们保护了 $\mathcal{SP}$ ，使得它不能获得任何两点之间的可达性信息，那么也就很明了的，它不能猜测出图中任何一条边的存在性，进而由于它不能获取到任何边的信息，所以它不能推断出图数据的任何结构信息。

**引理 4.3** 当且仅当攻击者 $\mathcal{SP}$ 要么可以破解单向防冲突哈希算法要么可以破解Elgamal加密算法，那么它才有可能判断图中一条边的存在性。

我们通过反证法来对这个引理进行证明。我们假设攻击者 $\mathcal{SP}$ 能够确定图中任何一条边 $(u, v)$ 的存在与否，那么攻击者 $\mathcal{SP}$ 必须至少获取到一个查询 $\text{Reach}(u, v)$ 的结果信息。在引理 4.2 中，我们证明了只有在 $\mathcal{SP}$ 能够破解了单向防冲突哈希算法或者Elgamal加密算法时，它才有可能获取到任何一个查询的结果信息。所以攻击者 $\mathcal{SP}$ 只有在它要么可以破解单向防冲突哈希算法要么可以破解Elgamal加密算法的情况下，那么它才有可能判断图中一条边的存在性。

**基于数据大小的攻击分析** 在上文中，我们针对攻击者可能进行的对密文进行攻击的情况进行了分析，在本节中，我们对基于集合大小的一些攻击手段进行分析。

**引理 4.4** 在算法 2 中，如果将 $\delta$ 设置为 0，那么查询点之间的可达性信息将完全能够防止 $\mathcal{SP}$ 基于集合大小(size-based)的攻击。

我们同样使用反证法来对这个命题进行证明，我们假设攻击者 $\mathcal{SP}$ 可以利用size-based方法猜测到查询点 $\text{Reach}(u_e, v_e)$ 的可达性信息。因此攻击者 $\mathcal{SP}$ 可以从以下几个方面来获得可达性信息：(1) $\text{Lout}^e(u_e) \cap \text{Lin}^e(v_e)$ 交集的大小；(2) $\text{Lout}^e(u)$  和  $\text{Lin}^e(v_e)$  的集合大小信息。由于对于任何两个集合的交集，在我们的算法 1 中，我们已经保证其交集的大小始终为 $I_{\max}$ ，同时，在算法 2 中，我们将 $\delta$ 设置为 0，那么对于任何的查询点都有 $|\text{Lout}^e(u_e)| = \text{Lout}_{\max}$ ， $|\text{Lin}^e(v_e)| = \text{Lin}_{\max}$ 。所以对于任何两个不同的查询， $\mathcal{SP}$ 获取的基于大小的信息都是完全一样的，它无法通过集合的大小信息获取到任何对攻击有用的信息。

**引理 4.5** 在算法 2 中，如果将 $\delta$ 设置为 0，就可以防止基于集合大小(size-based)的攻击方法对图结构信息的攻击。

这一命题的证明和引理 4.4类似。

在实际情况中， $\delta$ 不必要严格的位置为0，因为我们利用算法向 $\text{Lin}^s$ 和 $\text{Lout}$ 中加入了新的人工代理节点，使得 $\text{Lin}^s$ 和 $\text{Lout}$ 集合的大小不能直接的反映图上一个点的连通性信息，即一个度较大的点，它的 $\text{Lin}^s$  和 $\text{Lout}$ 集合大小不一定也比较大，所以 $\delta$ 不一定非要设置为0，在一个较小的范围内， $\delta$ 对安全性的影响不会非常的大，关于 $\delta$ 究竟对安全信息泄露多少信息的证明，我们在这里略去它的分析。

## 4.6 实验结果

在本节中，我们通过实验验证我们提出的技术的效率以及我们对原始2-hop索引建立算法的优化效果。

**实验设置** 我们在一台普通的PC上进行了我们的实验，机器配置有一颗i3-2310 2.10GHz 的CPU，4 GB内存并安装有Windows 7操作系统。本节中所有的算法都是用C++进行实现的，其中改进的2-hop索引建立算法是由R.Bramandia[6]提供，我们在其代码的基础上优化和修改而来。本节中所使用的哈希函数( $h_{s_1}$  和 $h_{s_2}$ )是使用的160-bit SHA-1算法。加密算法是使用1024-bit Elgamal加密算法。

表 4.1: 合成数据集

$G$	$ V(G) $	$ E(G) $	$ E(G) / V(G) $
SYN-1	3073	37615	12.24
SYN-2	5651	15968	2.83
SYN-3	4880	27946	5.73

**实验数据集** 在实验中，我们使用了3个人工数据集(简写为SYN)和4个真实数据集。关于这些数据集的一些特性详见表 4.1和表 4.2。所有的合成数据集使用的合成算法是最近在图数据库领域比较流行的[48]文章中的合成器生成的合成数据

表 4.2: 真实数据集

$G$	$ V(G) $	$ E(G) $	$ E(G) / V(G) $
YEAST	2361	7182	3.04
ODLIS	2909	18419	6.33
ERDOS	6927	11850	1.71
ROGET	1022	5075	4.97

集。我们在合成器中，通过设置 $\alpha = 0.27$  和 $\beta = 10$  来控制图的大小和图的稀疏度。所有的真实数据集，我们都可以通过网络进行下载<sup>1</sup>。

**查询集** 对于任何的合成数据集和真实数据集，我们生成1000个随机的查询，其中50%的查询集是从它们的传递闭包集合中随机选择的可达的查询，另外50%是在传递闭包集合之外选择的不可达的查询。

**2-hop索引创建的启发式条件** 我们已经实现了最原始的[17]文章中的基于`maxDensCover` 的2-hop索引建立算法，同时我们也比较了在[15]中提出的基于 $\text{maxSetCover} = |E_w \cap T'|$  的启发式条件算法。以及我们在本文中提出的具有 $I_{\max}$ 感知的`maxISCover`启发式方法。这些启发式方法可以通过结合 3.1节中的2-hop索引创建流程来生成基于不同启发式条件的2-hop索引。同时在实验中，我们将 $\delta$ 默认设置为0。

表 4.3: 最大交集集合大小 $I_{\max}$ 比较

$G$	$I_{\max}$		
	<code>maxDensCover</code>	<code>maxSetCover</code>	<code>maxISCover</code>
SYN-1	2558	22	15
SYN-2	17	7	3
SYN-3	1169	48	13

#### 4.6.1 合成数据集上的实验

**maxISCover 的作用比较** 在表 4.3中，我们比较了在三种不同的启发式条件下，建立出的2-hop索引中的最大的交集集合大小 $I_{\max}$ 的比较。我们可以从表中很明显的看出，通过`maxISCover`启发式条件创建出的索引的最大交集集合大小 $I_{\max}$ 要比`maxDensCover` 和`maxSetCover`启发式条件要小很多。其主要原因，我们在利用`maxISCover` 启发式条件创建2-hop索引时，在每一次迭代中我们总是将最大交集集合的大小考虑到索引的建立过程中，所以每次我们总是选择能够使得 $I_{\max}$ 尽可能小的中心点加入到`Lin`和`Lout`中。从具体的实验数据来看，`maxISCover`平均要比`maxDensCover` 和`maxSetCover`的实验效果要好88倍和2.5倍。同时，我们看到一个有趣的现象，虽然`maxSetCover`启发式条件在每次选择时并没有考虑到选择对 $I_{\max}$ 的影响，但是它的结果也是比较好的，其主要原因在于`maxSetCover`每

<sup>1</sup>YEAST: <http://vlado.fmf.uni-lj.si/pub/networks/data/bio/Yeast/Yeast.htm>

ODLTS: <http://vlado.fmf.uni-lj.si/pub/networks/data/dic/odlis/Odlis.htm>

ERDOS: <http://vlado.fmf.uni-lj.si/pub/networks/data/Erdos/Erdos02.net>

ROGET: <http://vlado.fmf.uni-lj.si/pub/networks/data/dic/roget/Roget.htm>

次都选择能够覆盖最多未覆盖的传递闭包集合 $T(G)$ 的最多元素的中心点，所以这使得可能使用较少的中心点覆盖到所有的传递闭包集合的信息，通过我们的实验发现这有时候会使得 $I_{\max}$ 会相对较小，但是相比于我们的启发式方法，它的结果并不是最理想的。

表 4.4: 总共添加的代理节点数量比较

$G$	Naive vs. MASN		
	maxDensCover	maxSetCover	maxISCover
SYN-1	7.86M vs. 12.11K	67.61K vs. 17.30K	46.10K vs. 13.73K
SYN-2	96.07K vs. 8.75K	39.56K vs. 6.24K	16.95K vs. 6.08K
SYN-3	5.70M vs. 23.03K	0.23M vs. 18.92K	63.44K vs. 11.11K

**unifyIS算法效果** 接下来，我们比较交集大小归一化unifyIS算法在不同的 $I_{\max}$ 条件下的效果。在表 4.4 中我们展现了对于在不同的情况下，利用unifyIS算法(算法 1)总共添加的代理节点的数量和一个最朴素的添加代理节点的算法(Naive)比较。朴素的代理节点添加算法每次都选择一个新的代理节点添加到2-hop索引中，而不考虑重复利用之前已经使用过的代理节点。从表格中的结果中我们可以看出，在任何不同的启发式条件下，我们的MASN 添加的总共的代理节点数量总是至少比朴素的代理节点添加算法要少3倍左右。同时，我们注意到，在利用maxISCover启发式条件下，由于 $I_{\max}$ 一直都是最小的，然后我们在对比表中不同启发式条件下添加的代理节点的数量，可以看出，具有较小的 $I_{\max}$ 添加的代理节点数要比较大的 $I_{\max}$ 要少很多，所以说明我们的算法中，使得 $I_{\max}$ 尽可能小可以使得最终的索引大小也相对小很多，我们的索引方法在三类不同的启发式方法中，效果仍然是最优的。

表 4.5:  $\mathcal{SP}$  和客户端查询时间

$G$	服务端 $\mathcal{SP}$ (ms) vs. 客户端(ms)		
	maxDensCover	maxSetCover	maxISCover
SYN-1	106.54 vs. 0.52	2.55 vs. 0.43	2.02 vs. 0.46
SYN-2	2.01 vs. 0.56	1.37 vs. 0.67	1.79 vs. 0.47
SYN-3	52.35 vs. 0.54	4.44 vs. 0.52	2.15 vs. 0.52

**pp-2-hop查询效率以及系统吞吐量** 在表 4.5 中，我们列出了对于一个可达性查询在 $\mathcal{SP}$ 端和客户端所需要的时间，表中的所有时间，是1000个查询的平均时间。对于在 $\mathcal{SP}$ 端的时间，由于maxISCover启发式方法生成的 $I_{\max}$ 较小，所以 $\mathcal{SP}$ 端需要进行的基于密文的标志位的乘法次数要少很多，所以在所

表 4.6:  $\mathcal{SP}$  端吞吐量

$G$	$\mathcal{SP}$ (查询量/秒)		
	maxDensCover	maxSetCover	maxISCover
SYN-1	9	392	495
SYN-2	495	730	559
SYN-3	19	225	465

有的实验中，`maxISCover` 的查询时间是最好的。对于数据集 SYN-1 和 SYN-3，`maxISCover` 启发式方法要比 `maxDensCover` 效率高至少一个数量级，对于数据集 SYN-2，`maxISCover` 的查询时间要比 `maxDensCover` 快至少两倍。对于客户端的查询时间，由于所有的情况下，客户端只需要对  $R_e$  进行一次解密即可，而且对于任何的解密，解密程序几乎做相同数量的计算，所以对于所有的不同情况，在客户端的时间都是差不多相同的，并且都比较小。

在查询效率的基础上，我们在表 4.6 中计算了  $\mathcal{SP}$  端的吞吐量，计算结果表明，对于一台普通的计算机，对于使用 `maxISCover` 启发式方法建立的索引，服务端  $\mathcal{SP}$  可以提供每秒钟 500 次的查询，与其他的启发式方法建立的索引对比来看，`maxDensCover` 的吞吐量是最低的，虽然有时候 `maxISCover` 具有较好的吞吐量，但是它对数据集比较敏感，不同的数据集它的吞吐量的变化非常大。

#### 4.6.2 真实数据集上的实验

我们在四个真实的数据集上，进一步验证我们的算法在真实数据集上的效果。由于得到的结果和上面分析的合成数据集比较相似，我们在这里就展示一些比较有亮点的实验结果。

表 4.7: 最大交集集合大小  $I_{\max}$  比较

$G$	$I_{\max}$		
	maxDensCover	maxSetCover	maxISCover
YEAST	237	6	4
ODLIS	274	3	3
ERDOS	250	5	3
ROGET	752	6	4

表 4.7 中展示了真实数据集在不同的启发式方法下最大交集大小  $I_{\max}$  的大小情况。本文中提出的 `maxISCover` 相比于 `maxDensCover` 和 `maxSetCover` 都具有较小的  $I_{\max}$ 。由于 `maxISCover` 启发式算法产生的  $I_{\max}$  是最小的，进而从表 4.8 中可以看出，利用 `unifyIS` 算法添加的总的代理节点的数量也是三种不同的启

表 4.8: 总共添加的代理节点数量比较

$G$	Naive vs. MASN		
	maxDensCover	maxSetCover	maxISCover
YEAST	0.56M vs. 7.72K	14.17K vs. 2.81K	9.44K vs. 2.72K
ODLIS	0.80M vs. 8.39K	8.73K vs. 2.98K	8.73K vs. 2.98K
ERDOS	1.73M vs. 8.86K	34.64K vs. 7.03K	20.78K vs. 7.01K
ROGET	0.77M vs. 3.75K	6.13K vs. 1.40	4.09K vs. 1.28K

表 4.9:  $\mathcal{SP}$  和客户端查询时间

$G$	$\mathcal{SP}$ (ms) vs. Client (ms)		
	maxDensCover	maxSetCover	maxISCover
YEAST	12.38 vs. 0.58	0.89 vs. 0.50	0.73 vs. 0.47
ODLIS	13.33 vs. 0.68	0.59 vs. 0.50	0.65 vs. 0.53
ERDOS	11.98 vs. 0.55	1.34 vs. 0.59	0.97 vs. 0.52
ROGET	31.91 vs. 0.64	0.57 vs. 0.59	0.29 vs. 0.64

发式方法中最少的。服务端 $\mathcal{SP}$ 和客户端的查询时间见表 4.9，在服务端基于 $\text{maxISCover}$ 建立的索引，它的查询时间几乎总是要比 $\text{maxDensCover}$ 快至少一个数量级以上。同时，和在合成数据集上的结果类似，由于客户端仅仅需要解密一次，所以客户端的时间都基本一致。

## 4.7 本章小结

在本章中，我们提出了一种隐私保护的可达性查询的索引建立算法和查询算法。我们提出了一种新的建立2-hop索引的启发式算法，使得在建立2-hop索引在选择中心点时既可以考虑尽可能覆盖到尽可能多的传递闭包集合元素同时使得全局的 $I_{\max}$ 最小化，在此建立的2-hop索引的基础上，我们提出了交集大小归一化算法和集合元素大小归一化两个不同的算法以及对2-hop索引进行加密来建立一种可以保护图数据结构和查询的可达性信息的pp-2-hop索引。最后我们在pp-2-hop索引的基础上，提出了一种基于密文域的可达性查询算法，通过利用乘法同态加密的特性，我们进一步优化查询算法，使得客户端仅仅需要进行一次解密就可以获得查询结果。然后我们对本文提出的隐私保护的可达性查询算法进行隐私安全的分析，证明我们的方法是可以实现对图数据结构和查询点之间的可达性信息进行保护。最后，我们通过一系列实验，来评估了我们方法的效率和方法的可行性。

## 第五章 稀疏图上可达性查询隐私保护方法

由于对于稀疏图这一种比较特殊的图数据，图中绝大多数的顶点之间不可达，如果我们仍然使用在第4章中提出的pp-2-hop索引算法，我们需要添加的人工代理中心点将会非常多，致使最后的索引大小较大。在本章中，我们首先介绍一种针对稀疏图的基于最小交集归一化2-hop索引方法，详细介绍它的构建算法并针对该算法进行实验，证明该算法在保证了我们前面定义的隐私保护目标的前提下，相比原始的2-hop的索引大小的变化，以及查询时间的改变。

### 5.1 m-2-hop索引算法

在本节中，我们首先给出m-2-hop索引算法的定义：

**定义 5.1** 一个图 $G = (V(G), E(G))$ 的最小交集归一化2-hop(*Minimum Unified Intersection 2-hop, m-2-hop*)就像是定义 4.4中定义的2-hop索引结构，我们将其表示为 $\text{Lin}$  和 $\text{Lout}$ ，并且它具有以下的性质：

- $\forall u, v \in V(G)$ , 如果 $u \rightsquigarrow v$ , 那么 $\text{Lin}(v) \cap \text{Lout}(u) = \{(w, \text{true})\}$ , 否则如果 $u \not\rightsquigarrow v$ , 那么 $\text{Lin}(v) \cap \text{Lout}(u) = \{(w, \text{false})\}$ ;
- $\sum_{u \in V(G)}(|\text{Lin}(u)| + |\text{Lout}(u)|)$ 最小化。

从上面定义 5.1可以看出，在基于m-2-hop索引上的可达性查询将是一个简单的集合的交集运算，同时，交集的结果总是只有一个元素。对于给定一个查询 $\text{Reach}(u, v)$ ，如果 $\text{Lin}(v) \cap \text{Lout}(u) = (w, \text{true})$ 表明顶点 $u$ 可以到达顶点 $v$ ，否则如果 $\text{Lin}(v) \cap \text{Lout}(u) = (w, \text{false})$  表明顶点 $u$  不可以到达顶点 $v$ 。

**例子 6** 我们在此仍然使用图 3.1中的2-hop基础索引作为我们讨论的例子，基于图 3.1中的2-hop索引，它的一种可能的m-2-hop见图 5.1，在该m-2-hop索引中， $I_{\max}$ 的值为1。同时我们在图中使用粗体对真实的中心点标记，对于非粗体的点表示我们算法添加的代理中心点。很明显我们可以看出 $\text{Lin}(4) \cap \text{Lout}(0) = \{0\}$ ，则表明节点0可以到达节点4，然而由于 $\text{Lin}(4) \cap \text{Lout}(5) = \{12\}$ ，表示节点5不可以到达节点4。

$v$	$\text{Lin}(v)$	$\text{Lout}(v)$
0	<b>3, 7, 8, 10</b>	<b>0, 1, 3</b>
1	<b>1, 7, 8</b>	<b>0, 1, 10</b>
2	<b>1, 4, 7, 11</b>	<b>0, 4, 8</b>
3	<b>0, 7</b>	<b>0, 8, 11</b>
4	<b>0, 5, 9, 12</b>	<b>2, 5, 7</b>
5	<b>0, 2, 9</b>	<b>2, 7, 12</b>
6	<b>0, 2, 6</b>	<b>6, 7, 9</b>

图 5.1: 基于图 3.1 中的图数据的 m-2-hop 索引

### 5.1.1 m-2-hop 算法分析

从定义 5.1 中可以看出, 为了构建 m-2-hop 需要两种不同类型的人工代理中心点。(1) 使用真实的中心点去覆盖传递闭包集合中的所有可达的点对之间的信息; (2) 使用虚假的代理中心点去覆盖那些  $u \not\sim v$ 。从上面的叙述中我们可以看出通过添加真实中心点去覆盖所有的  $u \sim v$  的问题和添加虚假的代理中心点去覆盖  $u \not\sim v$  是相同的问题。所以对于如果构建一个图的 m-2-hop 索引, 我们可以将其归纳为一个问题: 如何通过添加最少的真实(虚假)中心点使得其可以覆盖所有的  $u \sim v$  ( $u \not\sim v$ ), 且  $|\text{Lin}(v) \cap \text{Lout}(u)| = 1$ 。

**引理 5.1** 构建一个图  $G$  的 m-2-hop 索引是一个 NP-hard 问题。

构建 m-2-hop 算法的难度可以通过最小集合覆盖问题(Minimum Set Cover problem, MSC) 转变而来。我们下面证明构建一个图的 m-2-hop 和 MSC 是一个非常相近的问题。考虑图  $G$  的所有可达的点对, 我们使用一个二分图  $B = (V, L, R, E)$ ,  $L(B) = R(B) = V(G)$  来表示图  $G$  的传递闭包信息  $T(G)$ 。 $L(B)$  和  $R(B)$  分别表示 Lins 和 Louts, 对于所有的点  $\forall u \in R(B), v \in L(B)$ , 如果顶点  $u$  可以到达顶点  $v$  ( $u \sim v$ ), 那么存在  $(u, v) \in E(B)$ 。

我们考虑 MSC 的一个实例  $\mathcal{U} = E(B)$ , 其中包含了图  $G$  的所有传递闭包信息。我们使用  $\mathcal{S}$  表示  $\mathcal{U}$  的一个子集, 并且  $\mathcal{S} = \{S | S = E(K)\}$ ,  $K$  是二分图  $B$  的一个同构子图, 并且  $\forall u \in R(K), v \in L(K), (u, v) \in E(B)\}$ 。那也就意味着子图  $K$  包含了整个  $T(G)$  的所有信息, 并且覆盖了所有  $E(K)$ 。进一步, 我们需要利用完全二分图来覆盖  $E(B)$  中的信息, 并且确保  $E(B)$  中的每一条边被覆盖到一次。对于集合  $\mathcal{S}$  中的每一个  $S (S \in \mathcal{S})$ , 它的权重  $\text{weight}(S) = |L(K)| + |R(K)| = |V(K)|$ , 因为  $E(K)$  的信息是由通过向每一个  $L(K)$  的 Lin 和  $R(K)$  的 Lout 中添加一个中心点来覆盖的。

MSC 问题的目标是找到一个集合  $\mathcal{S}' \subset \mathcal{S}$  使得: (1)  $\mathcal{U}$  能够被集合  $\mathcal{S}'$  完全覆盖, 并且相应的集合  $E(B)$  中的每一条边都被覆盖。为了得到最小覆盖, 集合  $E(B)$  中

的每一条边都仅被覆盖一次，也就意味着如果顶点 $u$ 可以到达顶点 $v$ ( $u \rightsquigarrow v$ )，那么 $|\text{Lout}(u) \cap \text{Lin}(v)| = 1$ 。 (2)同时希望 $\sum_{S \in \mathcal{S}'} \text{weight}(S)$ 能够最小化，也就意味着 $\sum_{u \in V(G)} (|\text{Lin}(u)| + |\text{Lout}(u)|)$ 被最小化。很明显，找到一个这样的集合 $\mathcal{S}'$ 和找到一个最小的m-2-hop索引完全一样。由于我们知道MSC问题是一个NP-hard问题，所以构建一个m-2-hop是一个NP-hard问题。

我们对上面的构建一个图的m-2-hop索引问题使用一个传统的MSC贪心算法来解决，该贪心算法与最优解之间的近似率为 $(1+1/2+\dots+1/|\mathcal{U}|)\text{OPT}$ 。首先，我们定义一个 $\mathcal{U}'$  来表示集合 $\mathcal{U}$ 中未被覆盖的元素，也就是 $\mathcal{U}' = \mathcal{U} - E(B)$ 。集合 $\mathcal{S}$ 中带有最大的 $\frac{|S \cap \mathcal{U}'|}{\text{weight}(S)}$ 值的集合 $S$ 通过不断的迭代从集合 $\mathcal{S}$ 中移除来覆盖 $\mathcal{U}'$ ， 并从 $\mathcal{U}'$  中将已覆盖到元素移除。当 $\mathcal{U}'$  中所有的元素都已经被覆盖到，那么算法终止。如何找到这样的一个集合 $S$ 等同于从 $B$ 中找到这样一个最大完全二分子图 $K$ ，因为：

- (1)  $\forall u \in R(K), v \in L(K), (u, v) \in E(B);$
- (2)  $\frac{|S \cap \mathcal{U}'|}{\text{weight}(S)} = \frac{|E(K) \cap \mathcal{U}'|}{|V(K)|} = \frac{|E(K)|}{|V(K)|}$  是最大化的。

其中根据定义 $S = E(K)$ ，我们可以得到 $|S \cap \mathcal{U}'| = |E(K) \cap \mathcal{U}'|$ ,  $|E(K) \cap \mathcal{U}'| = |E(K)|$ 。问题就转化为寻找密集子图问题，由于MSC构建问题需要利用到最大完全二分图来覆盖 $E(B)$ ，这就是寻找密集子图问题。已经证明，需要一个二分图的最大完全二分图也是一个NP-hard问题，所以我们的m-2-hop索引也是一个NP-hard问题。

### 5.1.2 基于启发式方法的m-2-hop构建算法

在本节中，我们提出一个贪心算法来寻找最大完全二分子图问题，我们将该算法和基于MSC的m-2-hop的构建算法合并在一起得到我们的基于启发式方法的m-2-hop构建算法，见算法 3。算法的输入为一个图数据 $G$ ，输出为计算出的m-2-hop索引(Lin和Lout)。在算法的第1-2行中，我们首先将Lins 和Louts 设置为空集，同时生成图 $G$ 的传递闭包集合 $T(G)$ 以及它的补集 $T(G)^-$ 。在算法的第3-4行中，分别利用上面计算出的 $T(G)$ ,  $T(G)^-$ 得到他们的二分图集合 $B$ 和 $B^-$ 。并且通过调用AddSurNode 程序在生成的 $B(B^-)$ 基础上对Lin和Lout中添加真实(虚假)的的中心点，并将其标识位 $f$ 设置为true(false)。在第5行中，将生成的Lins 和Louts返回。

算法 3中的第6-9行AddSurNode是MSC问题的一个经典贪心算法的二分图描述。对于每一次迭代，如果 $E(B)$ 中的信息还没有被完全覆盖到( $E(B) \neq \emptyset$ )，则从 $B$ 里面通过GreedyFndMaxBiK算法选择一个最大的完全二分图 $K$ 。同时生成一个新的中心点，并根据其 $f$ 设置相应的标志信息，然后根据 $(u, v) \in E(K)$ 的

信息，将该中心点加入到 $\text{Lin}(v)$ 和 $\text{Lout}(u)$ 中。在第9行中，将已经被覆盖到的 $E(K)$ 从 $E(B)$ 中移除。

算法3中的第10-14行，GreedyFndMaxBiK算法用来寻找 $B$ 的一个最大完全二分子图 $K$ 。GreedyFndMaxBiK算法首先初始化一个空的图 $K$ 。然后算法如果能够找到一个点的集合 $T$ ，使得对于 $T$ 中的每一个点 $u \in T$ ，可以和 $K$ 中的点构成一个完全二分子图。在算法的第12行中，从 $T$ 中选择一个具有最大的度的点作为候选点 $u_{\max}$ ，并将 $u_{\max}$ 加入到 $K$ 中，使其构成为一个更大的完全二分图。算法GreedyFndMaxBiK在再也找不到这样一个集合 $T$ 时结束，并将当前的最大完全二分图 $K$ 返回。这样返回的 $K$ 是当前最大的一个完全二分图。使用GreedyFndMaxBiK的一个好处在于它的效率是最接近最优的：从 $B$ 中寻找一个最大完全二分子图等同于从 $B^-$ 中寻找一个最大独立子集 $M$ ，其中 $B^-$ 是 $B$ 的一个补图。众所周知，这样的一个贪心算法，找出的最大独立子集的近似度在 $1/(1 + \Delta)$ 。

---

**Algorithm 3 MUIS ( $G$ )**


---

**Input:** 一个图数据 $G = (V(G), E(G))$

**Output:** 图 $G$ 的m-2-hop Lin 和 Lout

- 1: 对于图中所有的顶点 $\forall u \in V(G)$ ,  $\text{Lin}(u) = \text{Lout}(u) = \emptyset$
- 2: 生成图 $G$ 的 $T(G)$  和 $T(G)^-$
- 3: 生成 $T(G)$ 的 $B$ ，并调用子程序AddSurNode ( $B$ , Lin, Lout, true)
- 4: 生成 $T(G)^-$ 的 $B^-$ ，并调用子程序AddSurNode ( $B^-$ , Lin, Lout, false)
- 5: **return** Lin 和 Lout

**Procedure 3.1 AddSurNode ( $B$ , Lin, Lout,  $f$ )**

- 6: **while**  $E(B) \neq \emptyset$  //边还没有被覆盖完
- 7:    $K \leftarrow \text{GreedyFndMaxBiK} (B)$
- 8:    $\forall (u, v) \in E(K)$ , 加入一个新的点 $w$ 到 $\text{Lin}(v)$ 和 $\text{Lout}(u)$ 中  
 $\text{Lin}(v) \leftarrow \text{Lin}(v) \cup \{(w, f)\}$ ,  
 $\text{Lout}(u) \leftarrow \text{Lout}(u) \cup \{(w, f)\}$
- 9:    $E(B) \leftarrow E(B)/E(K)$      从 $B$ 中移除 $K$ 中的边

**Procedure 3.2 GreedyFndMaxBiK ( $B$ )**

- 10: 初始化一个空的最大完全二分图 $K$
  - 11: **while**  $T \neq \emptyset$ ,  $T = \{u | \{u\} \cup V(K) 在 B 中构成一个最大完全二分图\}$
  - 12:   设置 $u_{\max} = \arg \max_{u \in T} (\text{Deg}(u))$
  - 13:    $V(K) \leftarrow V(K) \cup \{u_{\max}\}$
  - 14: **return**  $K$
-

## 5.2 基于 $m$ -2-hop的可达性查询的隐私保护算法

在本节中，我们定义了 $m$ -2-hop的加密算法以及如何基于加密的 $m$ -2-hop进行可达性查询的处理，最后我们对我们的隐私保护的可达性查询算法进行分析。

**定义 5.2** 一个图 $G$ 的隐私保护 $m$ -2-hop( $ppm$ -2-hop)就是图 $G$ 的 $m$ -2-hop索引方法的一个加密的索引，我们将其表示为 $\text{Lin}^e$  和 $\text{Lout}^e$ ，对于 $m$ -2-hop中的每一个中心点 $(w, f)$ ，我们通过加密算法将其变换为 $(h_{s_1}(w), \text{Enc}(f))$ ，其中 $h_{s_1}$ 是一个单向哈希函数， $\text{Enc}$ 是一个随机加密算法。

**隐私保护的可达性查询。** 基于以上的 $ppm$ -2-hop索引，可达性查询算法可以分为以下三个步骤：(1)客户端将 $\text{Reach}(u, v)$ 可达性查询通过哈希方法，使其转化为 $\text{Reach}^e(u_e, v_e)$ ，其中 $u_e = h_{s_2}(u)$ ,  $v_e = h_{s_2}(v)$ ，并同时将哈希后的可达性查询提交至服务端 $\mathcal{SP}$ 。(2)服务端 $\mathcal{SP}$ 检索出 $\text{Lin}^e(v_e)$ 和 $\text{Lout}^e(u_e)$ 并进行交集运算 $\text{Lin}^e(v_e) \cap \text{Lout}^e(u_e)$ ，交集的结果会得到一个加密的二元组 $(w_e, f_e)$ ， $\mathcal{SP}$ 从二元组中提取出 $f_e$ ，然后将该 $f_e$ 返回给查询的客户端；(3)客户端通过使用从数据拥有者获得的授权密钥对 $f_e$ 进行解密，得到最终的查询结果，如果解密结果为 $true$ ，则说明 $u \rightsquigarrow v$ ，否则说明 $u \not\rightsquigarrow v$ 。

**ppm-2-hop隐私保护分析。** 我们接下来，对本章中针对稀疏图数据提出的 $ppm$ -2-hop安全性进行分析，我们可以证明，我们的索引和查询方法在 3.3 节中定义的攻击模型情况下是完全安全的。

**引理 5.2**  $ppm$ -2-hop索引及其查询算法可以有效的保护查询顶点之间的可达性信息 $\text{Reach}^e(u_e, v_e)$ 。

首先对于 $(w_e, f_e) \in \text{Lin}^e(u)$  (或者 $\text{Lout}^e(u)$ )，攻击者 $\mathcal{SP}$ 无法推测一个中心点 $w_e$ 是否为真实的中心点或者是一个虚假的中心点。其主要原因在于：(1)对于图中的顶点 $u \in V(G)$ 和所有 $ppm$ -2-hop索引中的中心点 $w$ ，我们是通过使用带有不同参数的哈希方法进行哈希，进而 $\mathcal{SP}$ 无法确定这两类顶点之间的关系，也就无法确定一个点是否为原图数据中的一个顶点。(2) $f_e$ 是通过使用 $\text{Enc}$ 进行了加密，通常情况下我们可以使用AES加密算法对其进行加密，由于AES加密算法， $\mathcal{SP}$ 无法对其进行破解，所以 $\mathcal{SP}$ 无法获取到 $f_e$ 对应的原文信息。综上两点所述，攻击者既无法破解中心点的哈希算法也无法破解标志位的AES加密算法，所以它无法获取查询结果的任何信息。

基于上面的分析，同时，对于一个给定的查询 $\text{Reach}^e(u_e, v_e)$ ，攻击者 $\mathcal{SP}$ 无法对 $\text{Lin}^e(v_e)$ 和 $\text{Lout}^e(u_e)$ 进行破解，由于对于任何的查询 $\text{Lin}^e(v_e) \cap \text{Lout}^e(u_e)$ 的结果始终为只包含有一个二元组，它的大小始终为1。同时由于 $(w_e, f_e)$ 是被进行

表 5.1: 真实数据和合成数据集信息

$G$	$ V(G) $	$ E(G) $	$ V(G) / E(G) $	$ V(\text{DAG}(G)) $	$ E(\text{DAG}(G)) $
SYN-1	3,073	37,615	0.08	3,073	37,615
SYN-2	5,651	15,968	0.35	5,651	15,968
SYN-3	4,880	27,946	0.17	4,880	27,946
real-1	6,927	11,850	0.58	6,927	11,850
real-2	34,546	421,578	0.08	21,608	281,030
real-3	62,586	147,892	0.42	48,438	96,976
real-4	75,879	508,837	0.15	42,176	61,995
real-5	82,140	549,202	0.15	53,599	200,101
real-6	82,168	948,464	0.09	10,559	28,331
real-7	81,306	2,420,766	0.03	12,248	95,659

了加密，所以在查询处理期间，攻击者 $\mathcal{SP}$ 不能获取到查询的任何信息。所以查询的节点之间的可达性信息得到了保护。

**引理 5.3** ppm-2-hop索引及其查询算法可以有效的保护原始图数据的结构信息。

我们可以使用反证法来证明引理 5.3 的正确性。我们假设攻击者 $\mathcal{SP}$ 可以推测出图中的一条边 $(u, v)$ 的存在性，那么攻击者 $\mathcal{SP}$ 必然可以破解至少一个查询 $\text{Reach}^e(u_e, v_e)$ ，这个和引理 5.2 中 $\mathcal{SP}$ 不能破解任何ppm-2-hop索引相违背，所以ppm-2-hop索引及其查询算法可以有效的保护原始图数据的结构信息。

### 5.3 实验结果

在本节中，我们通过实验对ppm-2-hop中算法的效果进行实验室验证。

**实验设置** 我们在一台普通的PC上进行我们的实验，这台机器配置有一颗3.40GHz的CPU和16GB内存。我们使用7个真实数据集和3个合成数据集来验证我们算法的有效性。7个真实的数据集其中的real-1(ERDOS)来自于[1], real-2 (Cit-HepPh), real-3 (p2p-Gnutella31), real-4 (soc-Epinions1), real-5 (soc-sign-Slashdot090221), real-6 (soc-Slashdot0922)和real-7 (ego-Twitter)实验数据集来自于[2]。关于这些实验数据集的一些特性信息见表 5.1。对于每一个图，我们首先将其转化为一个有向无环图(DAG)后，再对其建立ppm-2-hop索引。

表 5.2: ppm-2-hop实验结果

$G$	$ ppm\text{-}2\text{-hop} $	$ T(\text{DAG}(G)) $	$ V(\text{DAG}(G)) ^2$	索引构建时间
SYN-1	2.11M	2.91M	9.44M	11s
SYN-2	7.30M	0.29M	31.93M	1min25s
SYN-3	9.24M	2.44M	23.81M	1min35s
real-1	1.49M	400K	47.98M	22s
real-2	19.13M	84.25M	466.90M	16min43s
real-3	11.34M	18.17M	2.35G	28min51s
real-4	12.02M	348.83M	1.78G	28min20s
real-5	20.33M	371.90M	2.87G	52min38s
real-6	434.15K	21.18K	111.49M	18s
real-7	606.77K	37.78K	150.01M	32s

**ppm-2-hop的效果比较** 在表 5.2中，我们列出了 ppm-2-hop 索引的效果数据信息。我们从结果中可以看出， $T(G)$ 中真实的中心点的数量约为 $|V(G)|^2$ 的9%左右，所以 ppm-2-hop 索引的大小都几乎是 $|V(G)|^2$ 级别的1/11左右，大小相比于之前 pp-2-hop 有了进一步的改善。同时，我们可以看到，针对真实数据集，ppm-2-hop 的索引大小要比 $|V(G)|^2$ 的4%左右，在真实的数据集上，我们的算法的效果更加优秀。ppm-2-hop 索引的创建时间，最大的在53分钟，这个时间相比图的大小来说也是一个能够接受的范围，但是如果我们使用[14]的算法，该创建时间会大大缩短。

**查询效果** 我们从这些图中随机选择1000个查询点对。对于所有的查询，其服务端 $\mathcal{SP}$ 的查询时间都非常的快，对于合成数据集该查询时间在1.4s左右，对于真实数据集，该查询时间最大在0.18s，这些时间对于一个查询，都是一个可以接受的时间范围。

## 5.4 本章小结

在本节中，我们针对稀疏图数据提出了一种特殊的2-hop索引来提供一个可以实现隐私保护的可达性查询索引结构和查询算法。在本节中，我们提出了一种基于最小顶点覆盖的启发式构建ppm-2-hop算法，我们通过实验表明，ppm-2-hop索引对于真实数据集和合成数据集都具有非常好的效果，同时，在本章中，我们也同样对我们提出的ppm-2-hop索引方法进行了隐私保护的证明，证明我们的索引结构可以实现可达性查询时的隐私保护功能。

## 第六章 总结与展望

在本文中，我们研究了可达性查询的隐私保护方法。根据不同的图数据特性，提出了两种新的索引算法。针对具有一般特性的图数据，提出了一种普遍适用的pp-2-hop索引算法，在该算法中，我们使用新提出的启发式方法maxISCover代替原始的2-hop中的启发式方法maxDensCover，使得对于所得的2-hop索引能够具有较小的 $I_{\max}$ 。在此基础上，通过使用交集大小归一化算法和Lin与Lout大小归一化算法，并结合索引加密处理方法，最终得到本文中的pp-2-hop索引结构。除此之外，本文提出了基于pp-2-hop索引优化的可达性查询算法。针对本文中提出的隐私保护可达性索引和查询算法的安全性进行了分析。最后针对本文算法的效率和性能进行了实验，证明了本文pp-2-hop索引算法和查询方法的高效性和可扩展性。

针对特殊的稀疏图数据，本文中提出了另一种ppm-2-hop索引方法。在稀疏图中，大部分的顶点之间是不可达的，如果将所有的交集大小设置成第一种索引算法中的 $I_{\max}$ 大小，那么建立出的索引则会相对较大。为了解决该问题，我们通过引入ppm-2-hop算法，在文章中我们对该索引方法和查询算法给出了详细介绍。同时对ppm-2-hop索引的安全性进行了详细分析，最后通过实验证明了ppm-2-hop算法针对稀疏图的良好的实验结果。

在未来工作中，我们期望能够进一步提高pp-2-hop算法中关于代理中心点添加算法的效果，同时由于2-hop对大的图数据的支持不是非常友好，我们可以考虑将sigmod 2013中的TF-labeling 算法应用到本文中的算法，进一步提高算法对大的图数据的扩展性。最后，由于原始的2-hop索引可以支持最短路查询，所以我们期望在未来的工作中使得本文的算法同样可以支持隐私保护的最短路径查询。

## 参考文献

- [1] Pajek datasets. <http://vlado.fmf.uni-lj.si/pub/networks/data/>, 2013.
- [2] Stanford large network dataset collection. <http://snap.stanford.edu/data/>, 2013.
- [3] R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. SIGMOD, 1989.
- [4] G. D. Bader, M. P. Cary, and C. Sander. Pathguide: a pathway resource list. *Nucleic Acids Research*, 34(suppl 1):D504–D506, 2006.
- [5] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Class-based graph anonymization for social network data. *Proceedings of the VLDB Endowment*, 2(1):766–777, 2009.
- [6] R. Bramandia, B. Choi, and W. K. Ng. Incremental maintenance of 2-hop labeling of large graphs. *TKDE*, 22(5):682–698, 2010.
- [7] J. Cai and C. K. Poon. Path-hop: efficiently indexing large graphs for reachability queries. *CIKM*, pages 119–128, 2010.
- [8] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou. Privacy-preserving query over encrypted graph-structured data in cloud computing. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 393–402. IEEE, 2011.
- [9] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou. Privacy-preserving query over encrypted graph-structured data in cloud computing. In *ICDCS*, pages 393–402, 2011.
- [10] L. Chen, A. Gupta, and M. E. Kurul. Stack-based algorithms for pattern matching on dags. In *Proceedings of the 31st international conference on Very large data bases*, pages 493–504. VLDB Endowment, 2005.

- [11] Y. Chen and Y. Chen. An efficient algorithm for answering graph reachability queries. In *IEEE 24th International Conference on Data Engineering(ICDE 2008)*, pages 893–902. IEEE, 2008.
- [12] J. Cheng, A. W.-c. Fu, and J. Liu. K-isomorphism: privacy preserving network publication against structural attacks. *SIGMOD '10*, pages 459–470, 2010.
- [13] J. Cheng, S. Huang, H. Wu, and A. Fu. Tf-label: a topological-folding labeling scheme for reachability querying in a large graph. *SIGMOD*, pages 193–204, 2013.
- [14] J. Cheng, J. X. Yu, X. Lin, H. Wang, and P. S. Yu. Fast computation of reachability labeling for large graphs. *EDBT*, pages 961–979, 2006.
- [15] J. Cheng, J. X. Yu, X. Lin, H. Wang, and P. S. Yu. Fast computing reachability labelings for large graphs with high compression rate. *EDBT*, pages 193–204, 2008.
- [16] B. Chor et al. Private information retrieval. *J. ACM*, 45:965–981, 1998.
- [17] E. Cohen et al. Reachability and distance queries via 2-hop labels. *SODA*, 2002.
- [18] S. Das, O. Egecioglu, and A. El Abbadi. Anonymizing weighted social network graphs. In *IEEE 26th International Conference on Data Engineering(ICDE 2010)*, pages 904–907. IEEE, 2010.
- [19] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1985.
- [20] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding regular expressions to graph reachability and pattern queries. In *IEEE 27th International Conference on Data Engineering(ICDE 2011)*, pages 39–50. IEEE, 2011.
- [21] Z. Fan, Y. Peng, B. Choi, J. Xu, and S. S. Bhowmick. Towards efficient authenticated subgraph query service in outsourced graph databases. *IEEE Transactions on Services Computing*, 99, 2013.
- [22] J. Gao, J. X. Yu, R. Jin, J. Zhou, T. Wang, and D. Yang. Neighborhood-privacy protected shortest distance computing in cloud. *SIGMOD*, pages 409–420, 2011.

- [23] X. He, J. Vaidya, B. Shafiq, N. Adam, and X. Lin. Reachability analysis in privacy-preserving perturbed graphs. WI-IAT, pages 691–694, 2010.
- [24] Informatics Outsourcing. Outsourcing Solution Service.  
<http://www.informaticsoutsourcing.com/>.
- [25] H. Jagadish. A compression technique to materialize transitive closure. *ACM Transactions on Database Systems (TODS)*, 15(4):558–598, 1990.
- [26] R. Jin, N. Ruan, S. Dey, and J. Y. Xu. Scarab: scaling reachability computation on large graphs. SIGMOD, pages 169–180, 2012.
- [27] R. Jin, N. Ruan, Y. Xiang, and H. Wang. Path-tree: An efficient reachability indexing scheme for large directed graphs. *TODS*, pages 7:1–7:44, 2011.
- [28] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 3-hop: a high-compression indexing scheme for reachability query. SIGMOD, pages 813–826, 2009.
- [29] R. Jin, Y. Xiang, N. Ruan, and H. Wang. Efficiently answering reachability queries on very large directed graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 595–608. ACM, 2008.
- [30] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. In *VLDB*, pages 1146–1157, 2011.
- [31] A. Kundu et al. How to authenticate graphs without leaking. In *EDBT*, pages 609–620, 2010.
- [32] A. Kundu et al. Efficient leakage-free authentication of trees, graphs and forests. *IACR Cryptology ePrint Archive*, page 36, 2012.
- [33] K. Liu and E. Terzi. Towards identity anonymization on graphs. SIGMOD, pages 93–106, 2008.
- [34] D. A. Menascé. Qos issues in web services. *Internet Computing*, 6(6):72–75, Nov. 2002.
- [35] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1st edition, 1996.
- [36] K. Mouratidis et al. Shortest path computation with no information leakage. *PVLDB*, 2012.

- [37] R. Schenkel, A. Theobald, and G. Weikum. HOPI: An efficient connection index for complex XML document collections. In *EDBT*, pages 237–255, 2004.
- [38] S. Seufert, A. Anand, S. Bedathur, and G. Weikum. Ferrari: Flexible and efficient reachability range assignment for graph indexing. *ICDE*, pages 1009–1020, 2013.
- [39] K. Simon. An improved algorithm for transitive closure on acyclic digraphs. *Theoretical Computer Science*, 58(1):325–346, 1988.
- [40] S. Trissl and U. Leser. Fast and practical indexing and querying of very large graphs. *SIGMOD*, pages 845–856, 2007.
- [41] S. Trißl and U. Leser. Fast and practical indexing and querying of very large graphs. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 845–856. ACM, 2007.
- [42] S. J. van Schaik and O. de Moor. A memory efficient reachability data structure through bit vector compression. *SIGMOD*, pages 913–924, 2011.
- [43] H. Wang, H. He, J. Yang, P. S. Yu, and J. X. Yu. Dual labeling: Answering graph reachability queries in constant time. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 75–75. IEEE, 2006.
- [44] H. Yildirim, V. Chaoji, and M. J. Zaki. Grail: scalable reachability index for large graphs. *PVLDB*, 3(1-2):276–284, 2010.
- [45] J. X. Yu and J. Cheng. Graph reachability queries: A survey. In *Managing and Mining Graph Data*, pages 181–215. Springer, 2010.
- [46] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *Privacy, security, and trust in KDD*, pages 153–171. Springer, 2008.
- [47] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515, 2008.
- [48] L. Zhu, B. Choi, B. He, J. X. Yu, and W. K. Ng. A uniform framework for ad-hoc indexes to answer reachability queries on large graphs. *DASFAA*, pages 138–152, 2009.

- [49] L. Zou, L. Chen, and M. T. Özsü. k-automorphism: a general framework for privacy preserving network publication. *VLDB*, pages 946–957, 2009.

# 学术论文

## 发表论文

1. **Shuxiang Yin**, Zhe Fan, Peipei Yi, Byron Choi, Jianliang Xu, Shuigeng Zhou. Privacy-Preserving Reachability Query Services. DASFAA 2014, Part I, LNCS 8421 proceedings: 203-219 (2014)
2. Peipei Yi, Zhe Fan, **Shuxiang Yin**. Privacy-Preserving Reachability Query Services for Sparse Graphs. Proceedings of the 5th International Workshop on Graph Data Management: Techniques and Applications (GDM '14), 2014
3. 尹树祥, 靳婷. 图数据隐私保护可达性查询算法研究. 计算机工程, 2015
4. 张一帆, 尹树祥. 准确安全的邻近检测方法. 计算机工程, 2015

## 致谢

三年的硕士研究生学习即将结束，在这三年里，我的家人，导师，身边的同学和朋友给了我莫大的帮助，在我最困难的时刻给我鼓励和帮助，在我迷茫彷徨时，给我指导和帮助，使我能顺利完成学业，步入社会。在此，我要感谢所有帮助过我的人。

首先，我要感谢我的导师周水庚老师。从大四进入周老师的实验室以来，至今已有三年多。周老师对我在学术研究中给予了我极大的帮助。周老师总能对一个课题前沿有深刻的见解，对科研工作要求严谨。周老师认真踏实的科研态度，对我的科研和学习生活留下了深刻的影响。同时，每次我遇到科研或者生活中的困难，周老师总能尽力帮助我。衷心感谢周老师这三年多以来的照顾和培养。

其次，我要感谢香港浸会大学的Byron Choi教授，Byron在科研给予了我很大的帮助，他总是很耐心的对于学术上的问题跟我进行深刻的讨论。在跟他科研交流的过程中，总能受到科研思维上的启发，他的睿智让我印象深刻。

再者，我还要感谢实验室这个大家庭的所有同学。跟实验室同学的交流，使我的学生生涯过得更加充实。

最后，我要感谢一直支持我学业的家人，是他们的支持，才能让我克服困难，顺利完成硕士研究生三年的学习。

## 论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其它机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

作者签名: 尹树祥 日期: 2014.6.3

## 论文使用授权声明

本人完全了解复旦大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。保密的论文在解密后遵守此规定。

作者签名: 尹树祥 导师签名: 尹树祥 日期: 2014.6.3