目录

I 介绍

最近几年见证了基于图形结构数据[1]，[2]的应用程序的繁荣，如在线社交网络，道路网络，网络图[3]，生物网络和通信网络[4]，[5]。因此，在阿卡迪亚（例如，GraphLab [6]，Pregel [7]和 TurboGraph [8]）和行业（例如，Titan，DEX 和 GraphBase）中都提出了许多用于管理，查询和分析海量图的系统。随着云计算的普及，图表所有者（例如，基于图形的服务的企业和初创公司）希望将他们的图形数据库外包给云服务器，这引起了对隐私的极大关注。增强数据隐私的直观方法是在将图形外包到云之前加密图形。然而，这通常以低效率为代价，因为在加密图形上执行操作非常困难。
最短距离查询是最基本的图形操作之一，其根据特定标准针对图形中的给定源对和目的地对找到最短距离。然而，在实践中，用户在执行最短距离查询时可以考虑多个标准[2]。以道路网络为例，用户可能想知道在预算内两个城市之间的总收费的最短距离。该问题可以由约束最短距离（CSD）查询来表示，该查询基于具有对其他标准的一个或多个约束的一个标准来找到最短距离。

在本文中，我们关注单约束 CSD 查询。 这是因为大多数实际问题可以表示为单约束 CSD 查询。 例如，通信网络上的这种查询可以返回从起始节点到终端节点的最小成本，具有路由延迟的阈值。 此外，多约束 CSD 查询通常可以分解为一组子查询，每个子查询可以被抽象为单约束 CSD 查询。 形式上，CSD 查询 1 是这样的：给定原点 s，目的地 t 和成本约束 θ，找到总成本 c 不超过 θ 的 s 和 t 之间的最短距离。

该领域的现有研究大致可分为两类。第一类主要关注未加密图的 CSD 查询问题[2]，[9] - [12]。然而，这些方法不能容易地应用于加密图形环境中，因为在没有加密图形的特殊设计的情况下，这些方法中所需的普通图形上的许多操作（例如，添加，复制和比较）不能成功执行。第二类旨在通过加密图形[1]，[13]实现最短距离（或最短路径）查询。它们通常采用距离 oracles（预言机），使得任何两个顶点之间的近似距离可以有效地计算，例如以次线性方式。这些方法的主要局限在于它们无法对基于云的加密图执行约束过滤。因此，它们不能直接应用于回答 CSD 查询。

受现有方案局限性的影响，本文的目标是设计一种实用的图加密方案，通过加密图形实现 CSD 查询。由于普通图上的 CSD 问题已经被证明是 NP 困难的[10]，现有的研究（例如，[2]）通常采用近似解，这保证了所得到的距离不超过 α 倍。最短距离（其中 α 是图所有者预定义的近似比率），受成本约束 θ 的影响。图形加密会使 CSD 问题更加复杂。因此，我们也专注于设计近似解决方案。

具体来说，本文介绍了 Connor，一种新颖的图加密方案，针对加密图上的近似 CSD 查询。 Connor 建立在一个安全的 2-hop 标记索引（2HCLI）上，这是一种距

离预言机，可以有效地计算图中任意两个顶点之间的近似距离[1]，[2]。 安全 2HCLI 中的图的顶点由特定的伪随机函数（PRF）加密。 为了在允许成本过滤的同时保护图属性的实际值，我们通过订单显示加密（ORE）[14]，[15]和稍微同态加密（SWHE）来加密成本和距离（在顶点对之间） [16]，分别。 基于 ORE，我们设计了一种简单但有效的基于树的密文比较协议，可以加速云端的约束过滤过程。

本文的主要贡献如下。

1）我们提出了一种新的图加密方案 Connor，它可以进行近似的 CSD 查询。 它拥有可以在几毫秒内回答 α -CSD 查询的计算效率。（高性能）

2）我们设计了一个基于树的密文比较协议，这有助于我们确定两个整数之和与其密码文本之间的另一个整数与受控披露之间的关系。 该协议还可以作为其他相关应用场景中的构建块。（同态）

3）我们对 Connor 进行了全面的安全性分析，并证明它实现了最新的安全定义，名为 CQA2-security [17]。 我们还实施了一个原型，并对现实世界的数据集进行了广泛的实验。 评估结果表明了该方案的有效性和有效性。（可证明安全）

II 相关工作

在云计算时代，安全与隐私成为云服务用户[18]-[23]关注的焦点。本文从两个方面对相关工作进行了简要的总结。 CSD 查询普通图和图的隐私保护。

A. CSD 查询普通图

约束最短距离/路径在平面图上的查询问题引起了人们的广泛关注。Hansen[9]提出了一种增广 Dijkstra 算法，用于无索引的精确约束最短路径查询。然而，这种方法带来了巨大的计算负担。为了提高查询效率，另一个解决方案[11]侧重于近似约束最短路径查询，这些查询也是无索引的。

Storandt[12]提出了用索引查询精确约束最短路径的最新解决方案，它使用一种称为收缩层次结构的索引技术加速了查询过程。这种方法仍然会导致不切实际的高查询处理成本。Wang 等人提出了一种求解大规模路网中近似约束最短路径的方法。该方法充分利用叠加图技术，在原图的基础上构造了一个尺寸比原图小得多的叠加图。因此，他们在覆盖图上建立了约束的标签索引结构，大大降低了查询成本。不幸的是，所有这些解决方案都只适合对未加密的图执行查询。

B.图形隐私保护

在过去的十年中，随着云计算范式的广泛采用，人们对图形隐私的关注也越来越多。Chase 和 Kamara[17]首先引入了图形加密的概念，他们提出了一些用于图形操作的结构，例如邻接查询和相邻查询。Cao 等人利用“过滤-验证”原理，定义并解决了云计算中加密图数据的隐私保护查询问题。他们事先建立了基于特征的图索引，然后选择有效的内积进行滤波。一些方法如[13]、[25]、[26]利用差分隐私技术对图形进行私隐查询，可能会遇到这种情况

弱的安全。然而，这些研究引入了高得令人望而却步的存储成本，对大型图表不实用。孟等人提出了三种计算效率较高的结构，支持使用距离神谕查询近似最短距离，并且在半诚实的云服务器上证明是安全的。

安全多方计算(SMC)技术已广泛应用于解决保护隐私的最短路径问题[27]-[30]以及其他安全计算问题[31]。Aly 等人针对一般多方计算环境下传统组合图的最短路径问题，提出了两种图中最短路径的安全计算协议。Blanton 等人设计了数据无关算法来安全解决单源单目标最短路径问题，在稠密图上获得了最优或接近最优的性能。Keller 和 Scholl[29]为 SMC 设计了几种无关数据结构(例如优先队列)，并利用它们计算一般图上的最短路径。Gupta 等人提出了一种基于 smc 的方法来寻找符合策略的路径，这些路径的路由成本最低，或者满足不同网络域之间的带宽需求。然而，现有的通用 SMC 解决最短路径问题的方案可能会导致沉重的通信开销。

虽然对加密图的图查询已有相当多的研究，但是保护隐私的 CSD 查询仍然没有得到解决。本文提出了一种新颖高效的 CSD 查询图加密方案。

据我们所知，这是第一个能够对加密图形进行近似 CSD 查询的工作。 本文的其余部分安排如下。我们总结了第二节中的相关工作，并描述了第三节中近似 CSD 查询的背景。 我们在第 IV 节中正式定义了隐私保护的近似 CSD 查询问题。 之后，Connor 的构建在第 V 节中介绍，第 VI 部分详细描述了基于树的密文比较协议。 我们在第 VII 节展示了复杂性和安全性分析，通过第 VIII 节中的广泛实验评估了所提出的方案，并在第 IX 节中总结了本文。

III. 背景

本节给出了 CSD 查询问题的正式定义，并介绍了用于图形查询的 2HCLI 结构。

A.近似 CSD 查询

设 G = (V, E)为顶点集 V 和边集 E 的有向图（如果没有特别声明，后文提到的图即为有向图）。每条边 e ∈ E 对应距离 d(e) ≥ 0 和开销 c(e) ≥ 0.

我们把开销 c(e)作为约束条件。我们把连接两个顶点的一组边表示为一条路径。对于路径

P= (e1, e2,..., ek)，它的距离 d(P) 定义为 d(P)= $\sum_{i=1}^{k} d(e_i)$，即从起点到终点的距离。类似的，我们定义 P 的开销为 c(P) =$\sum_{i=1}^{k} c(e_i)$.本文中的符号在表 I 中进行了总结。

Notation	Meaning
$G = (V, E)$	Input graph
n, m	Number of vertices and edges in G
$d(e), c(e)$	Distance and cost of an edge e
$d(u, v), c(u, v)$	Distance and cost of the edge from u to v
$s, t, \alpha, \phi, \theta$	Origin, destination, approximation ratio, amplification factor and cost constraint in an α-CSD query
$\Delta, \widetilde{\Delta}$	Plain and encrypted graph index
$\Delta_{in}(v), \Delta_{out}(v)$	In- and out-label set associated with vertex v
d_θ	Depth of a cost constraint tree
β	Length of a path code
$E(m)$	ORE ciphertext of m
λ	Security parameter
k	Output length of ORE encryption
z	Output length of the SWHE algorithm
$\tau_{s,t}$	Query token
Y	Candidate sets as the outputs of the cost constraint filtering
$\mathcal{B}$	Maximum distance over all the sketches

表 I 符号列表

给定一个图 G，起点 s ∈ V，终点 t ∈ V ，且开销限制为θ，CSD 查询就是找到 s 和 t 之间的最短距离 d，并且总开销不超过θ. 由于 CSD 查询问题已被证明是 NP-hard，因此我们与已有的解决方案保持一致，本文重点提出一个近似的 CSD 解决方案。

受未加密图上的近似最短路径查询的通用定义的启发，我们定义近似 CSD 查询（即α-CSD 查询）如下。

定义 1（α-CSD 查询）：给定起点 s 和终点 t，开销限制θ和近似度α，一个α-CSD 查询返回路径 P 的距离 d(P)，使得 $c(P) \le \theta$ 且 $d(P) \le \alpha \cdot d$（opt），d（opt）是一个精确 CSD 查询的最优解，这个精确 CSD 查询具有同样的起点 s，终点 t 和限制θ.

图 1 显示了一个简单的图，它有五个顶点，每条边的距离和开销都在它旁边标记。设起点为 a，终点为 c，开销限制θ=4，精确 CSD 查询返回的最短距离 d（opt） = 6，对应的路径是 s (a, b, c). 对于近似度α = 1.5 的α-CSD 查询（参数不变，即起点为 a，终点为 c，θ=4），其中一个有效解是 8，对应路径为 Pα = (a, e, b, c).这是因为 $d(P\alpha) = 8 < \alpha \cdot d$（opt） = 9 且 $c(P\alpha) = 3 < \theta$.

基于上述定义，对于相同的起点和终点，给定两个路径 P1 和 P2，如果 $c(P1) \le c(P2)$ 且 $d(P1) \le \alpha \cdot d(P2)$，我们称 P1 α优于 P2.利用这一原理，我们可以显著降低图索引的构造复杂度，因为索引中的大量冗余项可以被过滤掉。我们将在下面的小节中作进一步的说明。

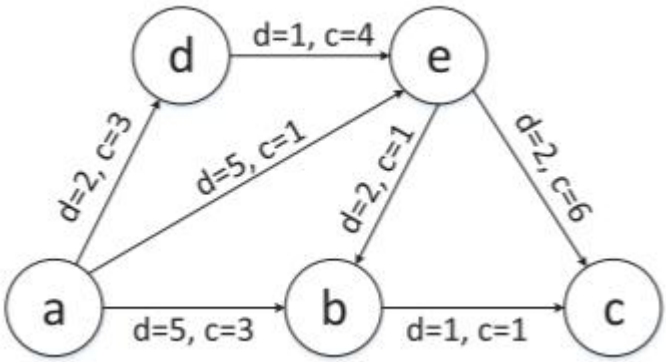

图 1. α-CSD 查询图示例

B.构建标签索引

本文设计的加密索引主要是基于众所周知的 2HCLI 构造的，2HCLI 是一种特殊的数据结构，能够有效地支持最短距离查询。现在我们简要介绍 2HCLI 的基本思想，并说明它在构建受限标签索引中的应用。

给定图 G = (V, E)，顶点集为 V，边集为 E，任意顶点 v ∈ V 都跟一个入标签集△in(v)和出标签集△out(v)相关联。

△in(v)中的每个实体对应顶点 u 到 v 的最短路径，u ∈ V.这意味着从 u 到 v 有一条或多条通路，但 v 不一定是 u 的邻居或两跳邻居。类似的，△out(v)中的每个实体代表从 v 到 V 中另一顶点 u 的最短路径。为了回答从起点 s 到终点 t 的最短距离查询，我们首先在标签集△out(s)和△in(t)中找出公共点，然后找出 s 到 t 的最短距离。注意△in(v)和△out(v)中的实体必须认真选取，以确保任意两个顶点 s 和 t 之间的距离都可以通过△out(s)和△in(t)计算出来。

考虑图 1 中的图，如果我们忽略边的开销限制，以 a 为起点，c 为终点，基本的无限制最短距离查询就可以通过 2HCLI 的帮助来解决，详见图 2。给定标签集△out(a)和△in(c)，就容易找出公共点的集合，里面包含点 b 和 e.这个基础最短距离查询的最终答案是 5，因为 d(a, e) + d(e, c) = 5 < d(a, b) + d(b, c) = 6.

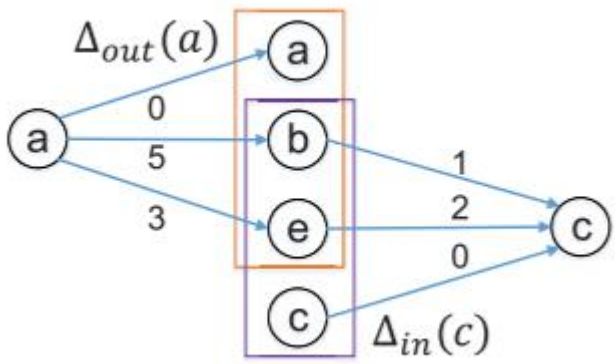

图 2. 一个基于 2HCLI 的基础最短距离查询示例。箭头旁边每个实体 d 表示从起点到终点的最短距离，如 a 到 e 的最短距离为 3。

虽然为只有距离条件的图构造 2HCLI 是简单而直接的，但是为 CSD 查询构造基于 2HCLI 的标签索引要复杂得多。这是因为在 CSD 查询中，边的限制条件有两种，因此在△in(v)和△out(v)标签集中，任意两点之间都可能有多种距离和开销的组合。为了便于说明，我们还以图 1 中的图和 CSD 查询为例。对应的 2HCLI 如图 3 所示，其中每个箭头旁边的二元组表示从起点到终点的距离和开销。注意，在图 2 的最短距离查询中，从 a 到 c 的最短距离是唯一的，就是经过 e 的那一条。然而，在图 3 所示的 CSD 查询设置中，按照不同的开销，通过 e 从 a 到 c 有四种可能的距离。由于开销条件的存在，在大尺度图中，每对顶点之间距离的数目可能显著增加，这使得构建 2HCLI 和解决 CSD 查询问题变得更加复杂。

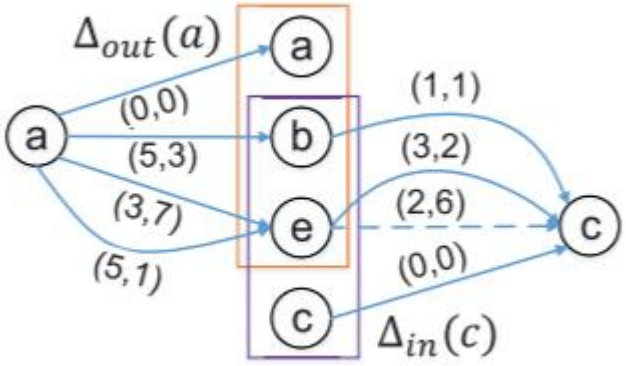

图 3. 精确 CSD 查询的 2HCLI 示例。箭头旁边的每个实体(距离，开销)分别表示距离和开销。开销限制θ = 4 时，从 a 到 e 的最短距离是 5.

为了提高查询效率，我们采用了离线过滤和在线过滤相结合的方法。

离线过滤的目的是降低 2HCLI 的结构复杂度，并尽可能地减少入标签集和出标签集中的实体数量。我们采用了[2]中提出的方法。2HCLI 中的实体是精心挑选的，使得对于任何从 u 到 v 且满足限制条件θ的 CSD 查询，都可以只用 2HCLI 就可以正确解答。由于在特定的 CSD 查询中，2HCLI 的结构应该独立于开销限制，我们可以使用“α优于”的定义去过滤入和出标签集中的冗余实体。

以图 3 为例，α = 1.5，从 e 到 c 的实体有两个，路径 P1ec = (e, b, c)，距离-开销二元组为（3，2），路径 P2ec = (e, c)，距离-开销二元组为（2，6），P1ec α优于 P2ec. 因此，P2ec 对应的实体可以被过滤掉（如虚线箭头所示），这有助于减少△in(c)中的实体数量。得到的 2HCLI 如图 4 所示。我们请读者参考[2]以获得更多的构造细节。

$\Delta(a)$	$\Delta_{in}(a)$	(a, 0, 0)
	$\Delta_{out}(a)$	(a, 0, 0), (b, 5, 3), (e, 5, 1), (e, 3, 7)
$\Delta(c)$	$\Delta_{in}(c)$	(b, 1, 1), (c, 0, 0), (e, 3, 2)
	$\Delta_{out}(c)$	(c, 0, 0)

图 4. 对图 3 中原始 2HCLI 进行离线过滤后得到的 2HCLI.实体(u, d, c)中各项分别表示顶点标识，距离和开销。这个近似 CSD 查询（起点为 a，终点为 c，α = 1.5, θ = 4）的解是 6，恰好是精确 CSD 查询的解。

在线过滤的目的是为给定的 CSD 查询选择可能有效的答案，并且仅基于 2HCLI。例如，给定从 a 到 c 的α-CSD 查询，限制条件θ = 4，我们可以先找到△out(a)和△in(c)的公共点集 V’。对于每个 v ∈V’，返回 c(a,v)+c(v, c) ≤ θ条件下 d(a,v)+d(v, c)的最小值。由于上述比较需要与相应的密文一起执行，因此在第六节将设计一种有效的在线过滤方法。

IV 问题公式化

本节介绍了系统模型和安全性保护隐私的模型α-CSD 查询,以及图加密方案的初步研究。

A 系统模型

我们采用通用系统模型在文献[1],[17]的保护隐私α-CSD 查询,如图 5 中所示,主要包括两种类型的实体,即一个用户和云服务器。

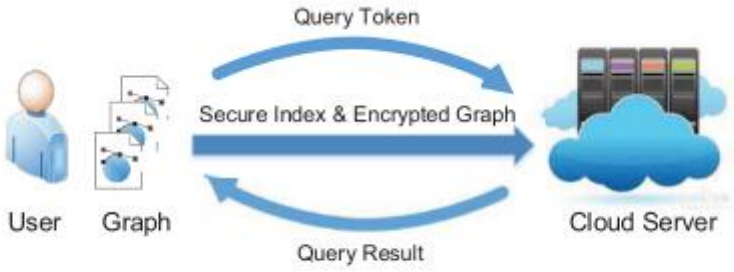

Fig. 5. The system model of privacy-preserving CSD query scheme.

用户为图构造安全的可搜索索引，并将加密的索引和加密的图一起发送到云服务器。当用户执行一个α-CSD 查询加密图,她首先生成一个查询令牌,然后提交到云服务器。接收到用户的查询令牌后，云服务器执行预先设计的查询算法，将安全索引中的条目与令牌匹配。最后,云服务器将答案回复用户的α-CSD 查询。

图形加密方案的正式定义如下。

定义 2(图加密):图加密方案Π = (KeyGen、Setup、Query)由三个部分组成多项式时间算法的工作如下:

(K, pk, sk)←KeyGen(λ):是一个概率的密钥生成算法，它将输入一个安全参数λ和输出一个密钥 K 和公共/秘密密钥对(pk, sk)。

△←设置(αK pk、sk、ϕ,G):是一种图像加密算法,它将输入一个近似比α,一个密钥 K,一个密钥对(pk, sk),一个放大系数ϕ和一个图 G,并输出一个安全索引△。

(dist q，⊥) ← Query((K, pk, sk, Φ, q)，△):是持有密钥 K、密钥对(pk, sk)和查询 q 的用户与持有加密图索引△的云服务器之间的两方协议。执行这个协议后,用户接收远程 dist q 作为查询结果，云服务器接收一个终止符⊥。

B 安全模型

图加密是对称搜索可加密(SSE)[34] -[38]的一种推广。因此在我们的图形加密方案中，我们采用了安全性定义的 SSE 设置。这个安全定义与最新的建议是一致的安全定义在[17]，[35]和[39]中也是已知的 CQA2-安全(即选择查询攻击安全性)。现在我们给出了 CQA2 安全的正式定义如下。

定义 3 (CQA2-安全模型): Let Π = (KeyGen,Setup, Query)是一种图形加密方案，考虑下面的概率实验，其中 A 是半诚实的对手，S 是模拟器，£ Setup 和£ Query 是(有状态的)泄漏函数。

1) Real (Π, A (λ)):

- 对手 A 输出一个图 G,近似比α和放大系数ϕ。
- 挑战者通过运行 Gen(1^{λ})来生成一个密钥 K 和公共/秘密密匙对(pk,sk)。通过设置(α，K ,pk、sk、ϕ,G),然后计算加密后的索引△，挑战者把加密索引△发送对手对手 A。
- 对手 A 进行多项式次的自适应查询，对于每个查询 q，对手 A 和挑战者执行查询((K, pk, sk, ϕ，q), △)。
- 对手 A 计算一比特 b∈{0,1}作为实验的输出。

2）Ideal (Π,A,S(λ)):

- 对手 A 输出一个图 G,近似比例α和一个放大系数ϕ。
- 给定泄漏函数£Setup (G)， S 模拟一个安全的图索引$△^{*}$，然后将它发送到对手 A。
- 对手 A 进行多项式次的自适应查询。对于每个查询 q，给出 S 的泄漏函数£query(G, Q)， 对手 A 和 S 模拟查询，其中对手 A 扮演云服务器的角色，S 扮演用户的角色。
- 对手 A 计算一比特 b∈{0,1}作为实验的输出。

我们说的图加密方案Π = (KeyGen,Setup, Query)是(£ Setup, £Query) -安全的，不受自适应选择查询攻击，如果所有对手 A 都存在一个 PPT 模拟器 S 那么:

$|Pr[Real\ \Pi, A\ (\lambda) = 1] - Pr[Ideal\ \Pi, A, S\ (\lambda) = 1]| \leq negl(\lambda)$,其中 negl(λ)是可忽略的值。

C 初步探究

现在，我们简要介绍一种在我们的设计中使用的加密技术，即 Order-revealing 加密。

Order-revealing 加密(ORE)是 order-preserving 加密(OPE)方案的推广，但提供了更强的安全保证。正如 Naveed 等人指出的[40]，OPE-encrypted 数据库极易受到推理攻击。为了解决这一局限性，提出了 ORE 方案 [14]，[15]，该算法是一个三元组Π= (ORE.Setup, ORE.Encrypt, ORE.compare) 描述如下:

- ORE.Setup(1^{λ}) → sk: 输入一个安全参数λ，然后输出密钥 sk。
- ORE. Encrypt(sk,m) → ct: 输入密钥 sk 和消息 m，输出密文 ct。
- ORE.Compute → z: 输入两个密文 ct 1 和 ct 2，输出 r∈{0,1}，表示对应的明文 m1 和 m2 之间的大于或小于关系。

Algorithm 1 Setup Algorithm for $GraphEnc_1$

Input: A secret key K, a key pair (pk, sk), an approximation ratio α, an amplification factor ϕ, and an original graph G.
Output: The encrypted graph index $\widetilde{\Delta}$.

Generate the *2-hop labeling index* $\Delta = \{\Delta_{out}, \Delta_{in}\}$ from G.
Initialize two dictionaries I_{out} and I_{in}.
Let $\mathcal{B}$ be the maximum distance over all the sketches and set $N = 2\mathcal{B} + 1$.
for each $u \in G$ **do**
  Set $T_{out,u} = h(K, u||1)$, $T_{in,u} = h(K, u||2)$.
  for each $(v, d_{u,v}, c_{u,v}) \in \Delta_{out}(u)$ **do**
    Compute $V = h(K, v||0)$.
    Compute $D_{u,v} = \text{SWHE.Enc}(pk, 2^{N-d_{u,v}})$.
    Compute $C_{u,v} = \text{ORE.Enc}(K, \phi c_{u,v})$.
    Insert $(V, D_{u,v}, C_{u,v})$ into the dictionary $I_{out}[T_{out,u}]$.
  end for
  Repeat the above procedure for each sketch in $\Delta_{in}(u)$ and add entries into $I_{in}[T_{in,u}]$.
end for
return $\widetilde{\Delta} = \{I_{out}, I_{in}\}$ as the encrypted graph index.

该算法：

输入是密钥 K，密钥对(pk,sk)，近似值比 α，一个放大系数 Φ，原始图 G。

输出是图加密索引。

主要流程是：

1. 从图 G 中生成 2-hop 标签索引：△out，△in。
2. 初始化这两个索引字典 I_{out}, I_{in}。
3. 定义 B 为图的最大距离，并设置 N = 2B+1;
4. 循环图中所有的顶点 u，并做如下操作：

   设置 $T_{out,u}$ = h(K,u || 1)， $T_{in,u}$ = h(K,u || 2)

   循环集合△out(u)中的二元组(△out(u)表示从 u 到 V 中另一顶点 v 的最短路径)，并做如下操作：

   计算 V = h(K,v || 0)

   计算 距离 Du,v = SWHE.Enc(pk, $2^{N-du,v}$)

   计算 花费 Cu,v = ORE.Enc(K, Φ $c_{u,v}$)

   把计算结果三元组插入到字典 Iout[$T_{out,u}$]

   按照上面步骤计算在△in（u）每个缩略图，并插入到字典 Iout[$T_{in,u}$]
5. 返回最终的图加密索引。

V、Connor 重建

在本节中，我们将介绍用于隐私保护 α -CSD 查询的图加密方案 Connor。

A. 重建简述

重建过程基于两个特定的伪随机函数 h 和 g，以及 SWHE 方案。在本文中，我们采用文献[16]中提到的 SWHE 方法。 关于 h 和 g 的参数信息，参考公式（1），

$$h: \{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda} \quad (1a)$$

$$g: \{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda + z + k} \quad (1b)$$

其中 λ 是加密参数，k 和 z 分别表示通过 ORE 和 SWHE 加密后的输出长度。

我们从简单的图 GraphEnc1 =（KeyGen，Setup，Query）开始，如下所示，包括：

•KeyGen：给定安全参数 λ，用户随机生成用于 SWHE 的密钥 K 和一对公钥和私钥（pk，sk）。

•Setup：给定原始图 G，近似比 α 和放大系数 ϕ，user 通过算法 1 可以计算得到加密的图索引. 其中，图 G 的 2HCLI $\Delta = \{\Delta_{out}, \Delta_{in}\}$可以通过第 III-B 节中描述的方法生成。

假设 B 是所有边的最大距离，并且 N = 2B + 1。受到文献[1]的启发，每个距离 $d_{u,v}$ 可以通过 SWHE 算法加密为$2^{N-d_{u,v}}$，以保护其真实值（第 8 行）。考虑到 $2^x + 2^y$受 $2^{\max(x,y)-1}$的限制，由 SWHE 加密的距离可以在一定数量的距离对上获得最小值。

对每个路径开销$C_{u,v}$，乘以放大系数 ϕ，由 ORE 算法加密（第 9 行）。 ϕ 一个很大的整数，应仔细选择，用以扩大$C_{u,v}$的空间。在实际应用中， ϕ 和所有连接边上的最大成本值的乘积应该足够大（例如，至少 2^{80}），其用于为输入提供足够的随机性选择。由于 ϕ 对用户来说是私有的，因此云服务器无法学习$C_{u,v}$的实际值。

•Query: 要执行具有原点 s，目的地 t 和花销约束 θ 的 α -CSD 查询，用户生成查询标记 $\tau_s = h(K, s||1)$ 和 $\tau_t = h(K, t||2)$，并将它们发送到云服务器。云服务器从索引获得 $I_{out}[\tau_s]$和 $I_{in}[\tau_t]$。对于出现在 $I_{out}[\tau_s]$和 $I_{in}[\tau_t]$中的每个加密顶点标识符 v，云服务器会采用一个花销约束策略（将在第 VI 节中详细描述），并将每对满足成本约束 ϕ θ 的节点对$(D_{s,v}, D_{v,t})$，添加到候选集 Y 中。需要注意的是，成本约束已经放大了 ϕ 倍，因为我们加密了 $\phi c_{u,v}$，而不是 $c_{u,v}$。

然后，云服务器直接获得$d = \sum_{i=1}^{|Y|} d_i$，其中对于 Y 中的每对$(D_{s,v}^i, D_{v,t}^i)$，$d_i = \text{SWHE.Eval}(\times, D_{s,v}^i, D_{v,t}^i)$。上述计算的正确性遵循 SWHE 的特性。可以参考文献[1]中的详细解释。

最后，云服务器将 d 返回给用户，用户使用密钥 sk 来解密 d，以获得 α -CSD 查询的答案。

请注意，这种简单的方法不仅可以正确回答加密图形上的 α -CSD 查询，还可以保护节点身份信息、距离和开销信息。

然而，对于算法 1 获得的已加密的图索引，在不执行任何查询的情况下，仍然会导致信息泄露。一方面，它揭示了每个加密图结构的长度，即$I_{out}[u]$和$I_{in}[v]$，以及所有图中 ORE 加密花销的信息。另一方面，它还公开了$I_{out}[u]$和$I_{in}[v]$之间的公共节点的数量，它表示将 u 连接到 v 的顶点的数量。特别是，如果云服务器知道没有公共顶点在$I_{out}[u]$和$I_{in}[v]$之间，它得知你无法达到 v。

图 2 的算法 2：设置算法

Input: A secret key K, a key pair (pk, sk), an approximation ratio α, an amplification factor ϕ, and an original graph G.
Output: The encrypted graph index $\widetilde{\Delta}$.

1: Generate the 2HCLI $\Delta = \{\Delta_{out}, \Delta_{in}\}$ of G.
2: Initialize two dictionary I_{out} and I_{in}.
3: Let $\mathcal{B}$ be the maximum distance over the sketches and set $N = 2\mathcal{B} + 1$.
4: **for** each $u \in G$ **do**
5: Set $S_{out,u} = h(K, u||1)$, $T_{out,u} = h(K, u||2)$, $S_{in,u} = h(K, u||3)$, and $T_{in,u} = h(K, u||4)$.
6: Initialize a counter $\omega = 0$
7: **for** each $(v, d_{u,v}, c_{u,v}) \in \Delta_{out}(u)$ **do**
8: Compute $V = h(K, v||0)$.
9: Compute $D_{u,v} = \text{SWHE.Enc}(pk, 2^{N-d_{u,v}})$.
10: Compute $C_{u,v} = \text{ORE.Enc}(K, \phi c_{u,v})$.
11: Set $T_{out,u,v} = h(T_{out,u}, \omega)$ and $S_{out,u,v} = g(S_{out,u}, \omega)$.
12: Compute $\Psi_{u,v} = S_{out,u,v} \oplus (V||D_{u,v}||C_{u,v})$.
13: Set $I_{out}[T_{out,u,v}] = \Psi_{u,v}$.
14: Set $\omega = \omega + 1$.
15: **end for**
16: Repeat the above procedure for each sketch in $\Delta_{in}(u)$ and obtain $I_{in}[T_{in,u,v}]$, except that: (i) set $T_{in,u,v} = h(T_{in,u}, \omega)$ and $S_{in,u,v} = g(S_{in,u}, \omega)$, and (ii) compute $\Psi_{u,v} = S_{in,u,v} \oplus (V||D_{u,v}||C_{u,v})$.
17: **end for**
18: **return** $\widetilde{\Delta} = \{I_{out}, I_{in}\}$ as the encrypted graph index.

B. 保护隐私的 α -CSD 查询

为了增强对敏感信息的保护，我们构建了一个隐私保护的 α -CSD 查询方案 Graph Enc2 =（KeyGen，Setup，Query），其中密钥生成过程与 Graph Enc1 相同，改进的索引构造和 CSD 查询过程，分别在算法 2 和 3 中进行了展示。

Graph Enc2 的 Setup 工作原理如下。用户首先构建图 G 的 2HCLI△，然后加密与 u∈G 相关的图（即，$\Delta_{out}(u)$ 和 $\Delta_{in}(u)$），如第 2-17 行所述。

请注意，为了防止出现先前提出的方法中关于图大小泄漏的问题，我们将每个加密图 $I_{out}[u]$ 和 $I_{in}[v]$ 分开，并确保它们分别存储在字典中，大小为 1。更确切地说，我们利用计数器 ω 并为 $\Delta_{out}(u)$ 中的每个个体生成唯一的 $T_{out,u,v}$ 和 $S_{out,u,v}$

（第 11 行）。同样，每个个体的唯一 $T_{out,u,v}$ 和 $S_{out,u,v}$ 也可以生成（第 16 行）。$T_{out,u,v}$（或 $T_{in,u,v}$）表示该个体将存储在 I_{out}（或 I_{in}）中的位置，这确保字典 I_{out}（或 I_{in}）中的每个位置仅具有一个个体。

$S_{out,u,v}$（或 $S_{in,u,v}$）用于与 $(V||D_{u,v}||C_{u,v})$ 进行 XOR 运算。由于 $S_{out,u,v}$（或 $S_{in,u,v}$）相比于每个图是不同的，因此 XOR 运算得到的 $\Psi_{u,v}$ 无法区分，这保证了静态加密图索引 $\tilde{\Delta}$ 既不泄露 I_{out} 和 I_{in} 之间的公共顶点数，也不泄露成本的花销信息。

算法 3 中的查询实现如下。假设用户想计算 s 和 t 之间的最短距离，其总成本不超过 θ 。她首先生成查询标记 $\tau_{s,t}$ 并将其发送到云服务器（第 1-3 行）。在接收到令牌 $\tau_{s,t}$ 时，云服务器在索引中搜索并获取 L_s 和 L_t（第 5-22 行）。也就是说，云服务器迭代地判断字典 I_{out} (I_{in}) 是否包含密钥 $T_{out,s,v}$ $(T_{in,v,t})$。如果存在，则将相应的个体添加到集合 L_s (L_t) 中。

一旦获得 L_s 和 L_t，云服务器就执行花销约束过滤算法（第 23 行），并计算 d（第 24 行），这与直接方法中描述方法相同。最后，用户利用秘钥 sk 可以从云服务器解密 d 后对应的答复。

图 Enc2 的算法 3 查询算法

Input: The *user*'s input are the secret key K, secret key pair (pk, sk), an amplification factor Φ, and the query $q = (s, t, \theta)$. The *cloud server*'s input is the encrypted index $\widetilde{\Delta}$.

Output: *user*'s output is $dist_q$ and *cloud server*'s output is $\perp$.

user generates $S_{out,s} = h(K, s||1)$, $T_{out,s} = h(K, s||2)$, $S_{in,t} = h(K, t||3)$ and $T_{in,t} = h(K, t||4)$.
user constructs a cost constraint tree T_θ based on $\phi * \theta$ using secret K as described in Section VI.
user sends $\tau_{s,t} = (S_{out,s}, T_{out,s}, S_{in,t}, T_{in,t}, T_\theta)$ to *cloud server*.
cloud server parses $\tau_{s,t}$ as $(S_{out,s}, T_{out,s}, S_{in,t}, T_{in,t}, T_\theta)$.
cloud server initializes a set L_s and a counter $\omega = 0$.
cloud server computes $T_{out,s,v} = h(T_{out,s}, \omega)$.
while $I_{out}[T_{out,s,v}] \neq \perp$ **do**
    cloud server computes $S_{out,s,v} = g(S_{out,s}, \omega)$.
    cloud server performs $(V||D_{s,v}||C_{s,v}) = \Psi_{s,v} \oplus S_{out,s,v}$.
    cloud server add $(V, D_{s,v}, C_{s,v})$ into L_s.
    Set $\omega = \omega + 1$.
    cloud server computes $T_{out,s,v} = h(T_{out,s}, \omega)$.
end while
cloud server initializes a set L_t and a counter $\omega = 0$.
cloud server computes $T_{in,v,t} = h(T_{in,t}, \omega)$.
while $I_{in}[T_{in,v,t}] \neq \perp$ **do**
    cloud server computes $S_{in,v,t} = g(S_{in,t}, \omega)$.
    cloud server performs $(V||D_{v,t}||C_{v,t}) = \Psi_{v,t} \oplus S_{in,v,t}$.
    cloud server add $(V, D_{v,t}, C_{v,t})$ into L_t.
    Set $\omega = \omega + 1$.
    cloud server computes $T_{in,v,t} = h(T_{in,t}, \omega)$.
end while
For each encrypted vertex identifier v that appears in both in L_s and L_t, the *cloud server* performs the cost constraint filtering operation through Algorithm 4, and add the pair $(D_{s,v}, D_{v,t})$ which satisfies the cost constraint $\phi\theta$ into a set Y. The pair that Algorithm 4 cannot verify is also added into Y.
For each pair in Y, the *cloud server* first computes $d_i = \text{SWHE.Eval}(\times, D^i_{s,v}, D^i_{v,t})$, and then computes $d = \sum_{i=1}^{|Y|} d_i$.
cloud server returns d to the *user*.
user decrypts d with sk.
return Decrypted value of d as $dist_q$.

VI TREE-BASED CIPHERTEXTS COMPARISON APPROACH

基于树的密文比较方法

A. Scenarios(情景)

假设有一个用户（U）和一个服务器（R）。U 有许多整数，这些整数先用一种加密算法加密，然后传给 R。 U 向 R 请求来获得一个和不超过 θ 的整数对（x，y）。注意，除了大于、相等或小于关系之外，x、y 和 θ 的明文不能向 R 公开。一种简单的方法是下载所有整数，在本地计算求和，并选择满足约束的整数对。然而，如果想要将这种计算方法加载到云端，这个方法就没有意义了。因此，应该找一个实际的办法来解决这一问题。

请注意，这种情形和众所周知的 SMC 方案是不同的。在 SMC 的设置当中，一组（两个或两个以上）具有私人投入的缔约方希望根据其投入来进行函数计算，同时除了函数的结果外什么都不显示。该函数用于许多实际应用，如外汇市场。SMC 是一个协作计算问题，解决了一组互相不信任的参与者之间的隐私保护问题。所有整数对（x, y）和约束条件（ θ ）的密文都上传到云服务器，云服务器来负责不等式测试。此外，我们还可以将两个密文的和与另一个密文之间的关系告诉服务器，这在文献[17]中被称为受控披露。

似乎我们可以利用同态加密技术，因为它支持计算 x+y 时的加操作，不过由于同态加密是概率加密，因此我们无法通过 x+y 和 θ 的密文来确定他们之间的关系。

B. Main Idea(主要思想)

基于树的密文比较协议的主要思想是用 ORE 原语对整数进行编码。根据我们目前的知识来看，没有方法能同时支持 ORE 和同态性质。因此，我们设计了一种新的方法来解决这个问题，这是由下列事实推动的：

如果我们想比较 x+y 和 θ，我们可以比较 x 和 θ /2 以及 y 和 θ /2。通过这两种对应关系的组合我们可以得到四种结果：

如果 x> θ /2（x≤ θ /2）和 y> θ /2（y≤ θ /2），我们可以知道 x+y> θ （x+y ≤ θ ）。在剩下的两个案例中，即 x> θ /2 和 y< θ /2，或 x≤ θ /2 和 y≥ θ /2，我们无法获获得确定性结果。这时，我们可以进一步将 θ /2 分为 θ /4。然后我们可以比较 x，y 与 θ /4 和 3 θ /4。

通过迭代执行这样的操作，我们可以越来越确定确定 x+y 和 θ 之间的关系。由于 ORE 性质，上述密文操作很容易便可以执行。接下来，我们将展示如何通过使用树结构有效地实现这个想法。C. Details of Protocol（协议细节）

为了实现通过密文来比较 x+y 和 θ，我们构造了一个成本约束树，它的节点表示与 θ 相关的特定值。为了清楚起见，我们定义 E（m）为 m 的 ORE 密文。

树结构的示例如图 6 所示。

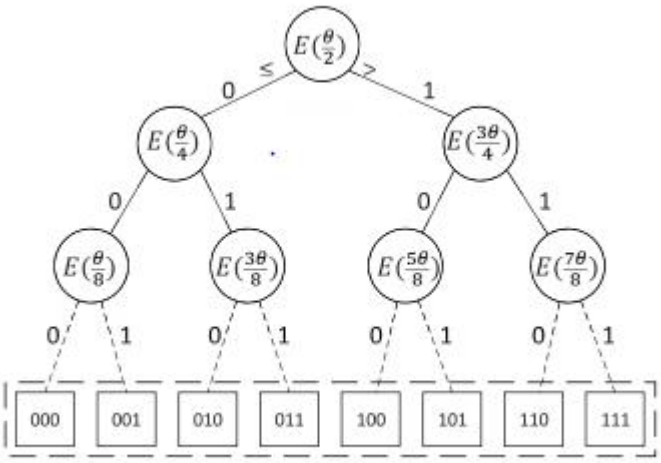

Fig. 6.

对于每个节点，我们将它的左子树定义为 0，右子树定义为 1。如果一个整数不大于该节点的值，则与左子路径进行进一步比较；否则，与右子树。因此，自根节点到叶节点的任一路径，我们可以得到一个路径代码，有效的显示了比较过程。例如，传入一个大小为 5θ/16 的整数，它将经过节点 E（θ/2）、E（θ/4）和 E（3θ/8），从而以代码 010 结束。我们定义了路径代码的长度（位数）为 β。β 实际上等于以 dθ 表示的树的深度。

现在我们可以通过如下过程来确定 x+y 和 θ 之间的关系。我们首先得到 x 和 y 的 ORE 密文，以及它们的路径代码 cx 和 cy（通过遍历树得到）。当计算 cx+cy 时，如果发生溢出（即 cx+cy≥2β），我们知道 x+y>θ。如果 cx+cy≤2β−2，我们也知道 x+y≤θ 否则，我们无法确定关系，最终以不确定性结束。

我们总结一下算法 4 中的这个过程

Algorithm 4 Tree-Based Ciphertexts Comparison Algorithm

Input: Two ORE ciphertexts $E(x)$, $E(y)$ and a cost constraint tree whose depth is d_θ.

Output: The relationship between $x + y$ and θ.

1: Initialize a counter $\omega = 1$ and two empty strings c_x and c_y.
2: **while** $\omega \leq d_\theta$ **do**
3: Visit the ω-th level of the tree with $E(x)$ and concatenate c_x with corresponding 0 or 1.
4: Visit the ω-th level of the tree with $E(y)$ and concatenate c_y with corresponding 0 or 1.
5: Set $\omega = \omega + 1$.
6: **end while**
7: **if** $c_x + c_y \geq 2^\omega$ **then**
8: **return** $>$.
9: **end if**
10: **if** $c_x + c_y \leq 2^\omega - 2$ **then**
11: **return** $\leq$.
12: **end if**
13: **return** *uncertainty*.