###### 七、复杂性和安全性分析

本节介绍了提出的图加密模型Connor的复杂性和安全性分析。

1. 复杂性分析

各过程复杂度主要影响因素分析：建立算法时间和空间复杂度：O (nμ)，n-顶点数，μ-总草图；

查询算法过程：查询令牌生成+云服务端CSD查询。时间复杂度：O(ηdθ)，空间负责度： O ()。 -G每个顶点关联最大值，dθ- ？）

Connor主要由建立和查询算法组成。

对图G生成的2HCLI进行加密 .让μ成为G中所有顶点的总草图,然后时间和空间复杂度都是O (nμ),其中n是G的顶点数。

查询算法包括

1. 用户端的查询令牌生成过程
2. 云服务器端的CSD查询过程。

Η是与G中的每个顶点关联的最大值。

1. 查询令牌生成过程的复杂度主要是由构建成本约束树决定，时间复杂度和空间复杂度都是O ()。
2. CSD查询过程,获取Ls和Lt、执行成本约束过滤，进行距离计算的时间复杂度分别是O(η)，O(ηdθ)和O(η)；上面三个组件的空间复杂度分别是O(η)，O(η+ 2 dθ) 和 O(η)。因此,CSD查询过程总的时间复杂度和空间复杂度分别是O(ηdθ)和O(η+ 2 dθ)

B .安全分析

接着介绍Connor的安全分析。首先说一下泄漏函数，然后证明Connor在CQA2 安全模型下是安全的。

1)设置泄漏

我们构造的泄漏函数LSetup揭示了从图G安全2HCLI中可以推导出的信息，包括图n中顶点总数，所有示意图上的最大距离

B = maxu∈vmax {(V, du, V,Cu、V)∈△out(V, du, V,Cu、V)∈△in}du, V,

和的大小。更准确地说，的大小由和中草图实体的总数组成，分别用和表示。因此，泄漏函数LSetup = (n,B, , )。

注意，成对成本的顺序关系和成本与成本约束之间的顺序关系不包括在LSetup中。每个实体的草图，加密后使用唯一的整数值进行XOR操作,这使每个实体在草图是无法区分的。

1. 查询泄漏

（个人理解：泄露函数组成：查询模式泄漏、草图模式泄漏、成本模式，分别将其形式化进行分析，模拟器S构造伪加密查询列表，敌手对于输出结果无法区分真假。）

泄漏函数LQuery由查询模式泄漏、草图模式泄漏、成本模式泄漏组成。分别来说，查询模式泄漏揭示了以前是否出现过查询，草图模式泄漏揭示了与查询顶点关联的草图、两个不同草图之间的公共顶点以及查询顶点的草图的大小。成本模式泄漏揭示了

1. 成本之间的顺序关系，
2. 查询过程中成本与成本约束之间的顺序关系。

我们将这些泄漏函数形式化如下。

定义4(查询模式泄漏):设q = (q1, q2，…， qm)是一个非空的查询序列。每个查询qi指定一个元组(ui、vi、θi)。

对于任何两个查询qi和q j,定义Sim(qi,qj) = (ui = u j, vi = v j,θi =θj), i.e., 即对于每个元素qj= (ui、vi、θi)是否成立。 然后，查询模式泄漏函数LQP (q)返回一个m×m(对称)矩阵，其中每个条目(i, j)等于Sim(qi, q j)。注意，LQP (q)不会泄漏查询顶点的标识。

定义5(草图模式泄漏):给定一个图G的安全2 HCLI 和查询q = (u, v,θ),素描泄漏模式函数LSP（，q)被定义为(∑,)。∑是一个列表,其中的每个元素都是与顶点相关联的草图,ϒ是一对(X, Z),其中X = h (v): (v,d,c)∈Iout和Z = h (v): (v d c)∈Iin是多重集合，h:{0,1}λ×{0,1}∗→{0,1}λ是一个特定的伪随机函数。

定义6(成本模式泄漏):成本约束查询中θ基本上可以由一定数量的均匀间隔构成。让dθ成本约束树Tθ(由第六节)的深度。与θ相关联的间隔((i−1)θ/ 2 dθiθ/ 2 dθ],当1≤≤2 dθ。分配每个区间μ列表。,第i个间隔和μi有关。 它存储这个区间的所有成本值。泄露间隔信息形式的Arr数组,其中第i个间隔是μi 元素(即。,Arr[i]=μi)。此外，假设z是被查询顶点的草图中的条目总数。对于每一对cost ci和cj，其大于、相等和小于的顺序关系分别可以表示为1、0和- 1。泄露的订单信息的成本是一个z×z(对称)矩阵∇每个条目(i, j)是1,0,或者−1。

因此,成本模式泄漏函数 Lcp (, q) = (Arr∇)。

因此，LQuery = (LQP(q)，LS P(， q)，LC P(, q))。

泄漏函数是在2HCLI上定义的，而不是在原始图上定义的。事实上，原始图的信息泄漏仅限于被查询的源-目标顶点的最小路径数。定义为一个n×n(对称)矩阵，其中n是图中顶点的个数。其中的每个元素为NULL、0或正整数，表示不确定状态(拓扑结构得到很好保护)、断开连接或两个查询顶点的最小路径数。

对于2HCLI成本值，引入一个使用者放大系数φ来扩大明文和密文空间。因此，服务器不能仅从泄漏函数LC P(，q)所显示的订单信息来推断真实的成本值。对于2HCLI中的距离值，使用SWHE加密来保护它们的真实值不受服务器的影响。

定理1:如果加密原语g、h、ORE和SWHE是安全的，那么所提议的图形加密方案= (K eyGen、Setup、Query)是(LSetup、LQuery)-在自适应选择查询攻击情形下是安全的。

证明:关键思想是构造一个模拟器S。给定泄漏函数LSetup和LQuery, S构造一个伪加密的2HCLI结构△∗= {I\*out, I\*in}和一个查询\*列表。如果对于所有的PPT敌手A来说，他们无法区分这两款游戏的真假，我们可以说我们的图形加密方案(LSetup,LQuery)是安全的，可以抵抗自适应的选择查询攻击。

### 3) 仿真 （个人理解：对于敌手攻击，先从已有查询结果查询是否存在，有就用以前的生成，没有就根据定理公式生成）

S处理每一个顶点的ui(1≤i≤n)，生成一个基于泄漏函数LSetup在2HCLI中假的I out。S随机选择wi ui n 1 wi =,和样品Ii←{0,1}λ和ηi←{0,1}λ均匀无重复。对于所有的0≤i< wi, S会采取以下步骤来模拟每个素描:S计算lwi = h (li wi)和ηwi = h(ηi wi), h是一个特定的伪随机函数。然后，它通过计算v∗= h(K∗，v||0)来加密ui草图中的每个顶点v，其中K∗是一个伪密匙。它随机生成两个整数d和c，并使用SWHE和ORE方案对2N - d (N = 2B + 1)和c进行加密，得到密文D\*和C\*。

让=ηwi⊕(V∗| | D∗| | C∗)。S将W\*存在Iout\*索引中，I\*out显示在索引外。也就是说，I\*out [lwi] =。类似地，S生成一个伪I∗in，最后得到伪2HCLI∗= {I∗out, I∗in}。

模拟q∗：给定泄漏函数LQuery = (LQP(q)，LS P(△,q)，LCP(q))， S模拟查询令牌如下。S首先检查被查询的顶点S和t是否出现在之前的任何查询中。如果s之前出现过，s就会将S∗out s和T∗out 设置为之前使用的值。否则,它会T∗out =li和S∗out,s =ηi一些以前未使用的li和ηi。然后记得ηi之间的协议。查询顶点t，S会采取相同的步骤:设置S∗in,t和T∗in,t,类似地,将t与所选ηi和li联系起来。

模拟一个假成本约束树T∗θ,S首先检查如果θ出现在查询之前任何查询。如果θ出现，此前,S设置T∗θ以前使用的值。否则,构造一个完整的基于二叉树的θ和带有随机生成密钥的ORE模型加密每一个树节点。S返回这个加密树作为T∗θ。

S模拟查询过程，如下所示。给定查询令牌(S∗out,T∗out, s, S∗in,t, T∗in,t, T∗θ), S首先检查先前查询查询结果是否存在。如果是，S返回以前用作查询结果的值。否则，S检查被查询的顶点S(或t)之前是否被查询过。如果查询顶点s在之前的查询中出现，s将Ls\*设置为以前使用过的Lsp(f, q)的∑值。否则，对于新出现的顶点s, s将执行以下步骤:用s生成草图,首先初始化一组Ls∗年代和计数器ω∗= 0。(细节我也不太懂)

然后,它迭代计算T∗out,s,v = h (T∗out,s,w∗)和S∗out,s,v = g (S∗out,w\*),并添加元组(V∗, D\*s,d, C∗s,d)到L∗s,直到I∗out(T∗out, s, d)不存在,在那里( V∗s,D\*s,v, C∗s,v) =I∗[T∗out, s, d]⊕S∗out,s,v。类似地，S获取顶点t的集合 Lt。在获得L\*s和L∗t, S执行成本约束过滤操作基T∗θ得到候选集Y∗。该定理由SWHE的CPA-security得到，也就是说，S基于Y\*执行SWHE计算并返回查询结果。

由于加密原语g、h、ORE和SWHE是安全的，所以伪的2HCLI结构\*和查询序列q与真实的无法区分。因此，对于所有的PPT敌手A来说，他们无法区分这两款游戏是真实的还是理想的。因此,我们有

|Pr(Real ii, A(λ)= 1)−Pr(Ideal,A, S(λ)= 1]|≤negl(λ), 其中negl(λ)是一个可忽略的函数。

第八章性能评估

本节通过对真实的数据集进行实验，从而对图形加密方案进行评估。

A.步骤

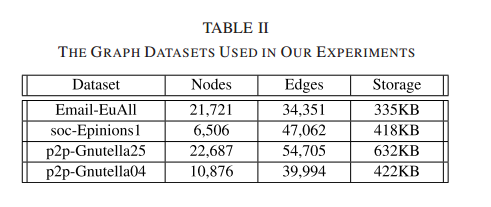
1.试验

我们实现了在 [2] 中介绍的构建 2HCLI 的方法。实现中的 ORE 和 SWHE 分别遵循 [15] 和 [16] 中描述的方法，此外，我们采用GMP库进行大整数运算。我们设置了安全参数 λ= 128 并且所有基本密码基元依赖OpenSSL库做支撑。我们实验中的所有算法都是用c++实现的。

硬件配置：Intel Xeon处理器，2.6 GHz, 8gb内存。

2.Graph 集合

实验中使用的数据集见表2。对于这些数据集，在有限的计算资源下随机选择它们的子集来构造可用索引。因为这些图是未加权的，我们为每条边生成一个长度和一个权重，其值服从均匀分布在1到100之间。采用成本作为约束条件。



3.方法比较

由于这是第一项解决加密图上CSD查询问题的工作，所以将这个方法与未加密图上的方法进行了比较。引入的最先进的纯文本图方法实现了比较。唯一的区别是在原始图上构造2HCLI，而不是覆盖图。本文的这个方法的实现具有更高的查询效率，当然索引构造的复杂性会更高。

4.查询集

针对每个数据集随机生成 200 个查询。每个查询中的源 s 和目标 t 也是随机选择的。另外，每一对 (s, t) 的成本约束 θ 的设置如下所述：用cmin表示从s 到t的所有路径的最小成本的下界,用cmax表示从s到t的最短路径的的最小花费的上界。如果成本约束 θ< cmin,那么将会没有有效的查询结果;如果成本约束 θ> cmax,那么最短的距离（shortest distance）将会是一直是有效的查询结果，为了减轻 θ 对性能的影响,为每个查询随机选择 50 个 θ 值,这些值得区间为[cmin, cmax]。

另一个重要的参数是 α ,它决定了近似保证 α-CSD查询。由于 α 对于所有查询是一个恒定常量,所以把它看作一个系统参数而不是特定查询的一部分。为了实现查询精度和系统效率之间的平衡,我们为所有查询设置了近似系数 α= 1.5

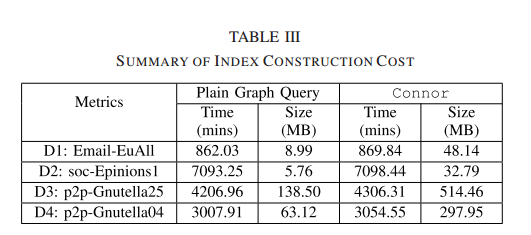
1. 安全2HCLI和查询令牌的评估

1.索引大小和构造时间

图的索引构造是一次性的并且是离线计算的，这个过程包括两个步骤:

1. 是构建无格式2 hcli,也就是与原始无格式CSD的索引构造相同,
2. 另一个是加密无格式2 hcli（这是本文的重点）

因此, ,我们考虑将第一步的输出作为未加密图的索引。索引大小和构建时间如表三所示。 不同数据集的索引大小和构建时间有很大的差异，这主要是由于图的拓扑的差异造成的。与原来最短距离查询中任意两个顶点之间只有一条最短路径不同，在CSD查询问题中，任意两个顶点之间通常存在多个约束最短路径。从直观上看，密集图可能比稀疏图具有更高的索引构建成本。



通常情况下，每个加密索引的大小大约比对应的普通索引大6倍。最重要的是索引的构建时间的比较：加密图略高于未加密图。因此，提高加密图上索引构建效率的关键是加快构建该图的无格式 2HCLI的过程。这个工作以后再做。

2.查询令牌生成

查询令牌的构造独立于特定的图形，我们现在分析查询令牌的大小和生成时间。查询令牌主要由5个元素,也就是： Sout,s, Tout,s, Sin,t , Tin,t , 和 Tθ. 前4个元素的长度都是16字节。**因为每个ORE密文的大小是16字节,一个深度为 dθ 成本树 Tθ 的大小为字节。因此,查询令牌的总大小是。由于 dθ 是一个相对较小的值，所以查询令牌的大小通常是小于1 KB。不同dθ的查询令牌生成时间如表4所示**。虽然查询令牌生成时间在 dθ 的影响下显著增加，但是一般情况下时间花费适中(例如,当dθ小于等于6)。

C.查询效率和准确性的评估

1)查询效率:评估查询效率,对于每个θ,生成成本约束*dθ*树有不同的深度。查询时间定义为从提交查询令牌到接收查询结果的时间间隔。我们计算了200个查询的平均查询时间。

平均查询时间与不同*dθ*加密2 HC LI是描绘在图7中,dθ从1增加到6。我们可以看到，**对于不同的图形数据集，查询时间变化很大**。对于每个数据集,提高*dθ*可以减少查询时间。这是因为大*dθ*可以过滤掉更多的距离对超过成本约束从而减少了在时间消耗上占主导地位的SWHE算法的候选距离计算量。

SHEN *et al.*: 基于云的加密图的近似CSD查询

图8所示、查询时间在平原2 HCLI加密2HCLI (dθ= 6)

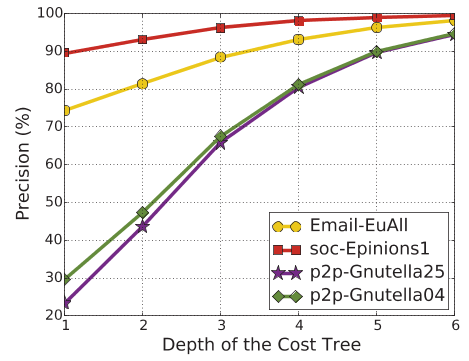


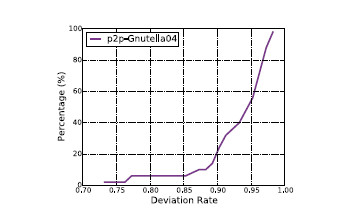
图9所示、查询精度的不同深度dθ成本树

图8给出了不同数据集的普通和加密场景下的查询时间。**加密后的2HCLI的查询时间要比普通2HCLI的查询时间长，这是因为加密后的2HCLI对密文的操作非常耗时(如成本过滤和距离计算)**。此外，这些操作的时间复杂度与表III中所列的图形索引的大小密切相关，这导致了图8中四个数据集的差异。

2)**查询精度:在Connor中，影响查询精度的因素有两个，即基于树的密文比较和距离计算**。前者可能会在候选集合Y中保留一些不满足成本约束的距离对，而后者利用SWHE的特性，基于Y中的所有候选，获得一个近似但不精确的最短距离。

我们使用众所周知的度量精度(P)来评估成本约束滤波过程的精度。*P* = ，其中*Tp*和*Fp*分别表示成本真正满足或超过成本约束的Y中距离对的个数。我们使用与上面介绍的相同的查询，并为每个查询计算P。最后,我们可以获得的平均精度P̄所有的查询。

图9给出了P̄查询精度和深度之间的关系的成本约束树*dθ*在不同的数据集。我们可以看到所有的数据集,**P̄随*dθ*,因为成本约束树更大深度*dθ*帮助我们检测约束违反有更高的概率**,在第六部分讨论。特别是, 当*dθ*= 6时，P̄超过94%以上所有的数据集。

图10所示、CDF实验组的偏差率不同的查询(*dθ*= 6)

为了评估最终查询结果的准确性，我们提出了一个名为偏差率的度量。设re和rp分别为Connor返回的查询结果和算法对对应的平面图返回的查询结果。然后,我们定义偏差率ξ= re / rp,这表明多远偏离rp。显然，接近1的偏差率描述了更准确的查询结果。

图10为数据集p2p-Gnutella04上偏差率的累积分布函数(CDFs)。我们可以看到,**ξ大于0.90查询结果的80%以上,并大于0.73是最糟糕情况**。因此，Connor能够在计算复杂度适中的情况下达到较高的精度。

第九节 结论

在本文中，我们介绍了Connor，第一种支持基于云的近似CSD查询的图形加密方案。特别地，我们提出了一种基于树的密码文本比较协议，用于控制成本约束过滤。安全分析表明，Connor可以实现CQA2安全。实现了一个原型，并使用实际的图形数据集评估了性能。评价结果验证了Connor的有效性。在未来的工作中，我们计划设计支持动态索引更新的技术。