中国科学: 信息科学 2021 年 第 51 卷 第 10 期: 1721–1737
SCIENTIA SINICA Informationis
论文

移动边缘计算中基于位置信息的安全 skyline 查询处理方法

王缵[1], 丁晓锋[1*], 周潘[2*], 田有亮[3], 金海[1]

1. 华中科技大学计算机科学与技术学院, 大数据技术与系统国家地方联合工程研究中心, 服务计算技术与系统教育部重点实验室, 集群与网格计算湖北省重点实验室, 武汉 430074
2. 华中科技大学网络空间安全学院, 武汉 430074
3. 贵州大学计算机科学与技术学院, 贵阳 550025
* 通信作者. E-mail: xfding@hust.edu.cn, panzhou@hust.edu.cn

收稿日期: 2020–12–22; 修回日期: 2021–02–20; 接受日期: 2021–04–06; 网络出版日期: 2021–10–13

国家自然科学基金 (批准号: 62172179, 61972448, 61772215) 资助项目

摘要 针对移动边缘计算下查询的效率和安全问题, 本文开展了面向位置信息的移动边缘计算安全 skyline 查询的研究. 首先, 提出了移动边缘计算场景下的安全 skyline 查询框架; 其次, 针对边缘服务器资源受限的特性, 设计了新颖且统一的轻量级安全索引结构; 然后, 考虑云边协同中的隐私问题, 提出了基于移动边缘计算的安全 skyline 查询协议. 安全性分析表明该协议在半诚实模型下是安全的. 同时, 实验评估发现其比现有协议具有更高的查询效率.

关键词 安全 skyline 查询, 位置信息, 移动边缘计算, 安全索引, 半诚实模型

1 引言

随着云计算的蓬勃发展 [1], 不断丰富的应用和海量信息导致云服务器的计算压力剧增, 一定程度上已无法满足用户的需求. 与此同时, 5G 技术进一步助力了移动边缘计算 (mobile edge computing, MEC) 的发展, 利用网络边缘侧的服务器, 应用可以快速响应用户的请求, 并缓解主干链路的通信负担和云服务器的负载压力, 但是将数据和服务下放至边缘服务器的方式也增加了隐私泄露的风险. 因此, 数据安全与隐私问题就显得尤为重要.

本文研究了 MEC 下基于位置信息的安全 skyline 查询方法. 由于用户一般频繁对附近区域的位置数据进行查询, 所以 MEC 下的位置服务比云计算下的更具合理性和实际意义. 图 1 展示了 MEC 下基于位置信息的安全 skyline 查询示例, 当客户端向最近的边缘服务器发送查询 $E(Q_1)$ (如查询距自己最近且价格最便宜的旅馆), 如果 $E(Q_1)$ 是查询该街道上满足查询条件的 skyline 点, 边缘服务器

引用格式: 王缵, 丁晓锋, 周潘, 等. 移动边缘计算中基于位置信息的安全 skyline 查询处理方法. 中国科学: 信息科学, 2021, 51: 1721–1737, doi: 10.1360/SSI-2020-0395
Wang Z, Ding X F, Zhou P, et al. Secure skyline query processing in mobile edge computing over location-based data (in Chinese). Sci Sin Inform, 2021, 51: 1721–1737, doi: 10.1360/SSI-2020-0395

Point	Price
$E(P_1)$	$E(4)$
$E(P_2)$	$E(5)$
$E(P_3)$	$E(5)$
$E(P_4)$	$E(3)$
$E(P_5)$	$E(6)$
$E(P_6)$	$E(7)$
$E(P_7)$	$E(6)$
$E(P_8)$	$E(4)$

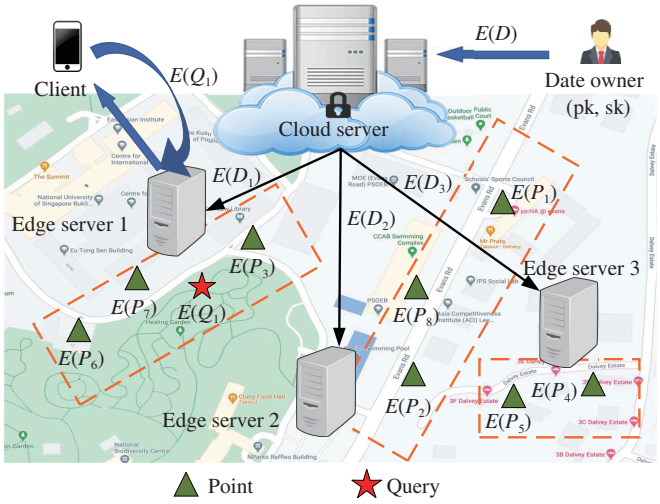

图 1 (网络版彩图) 移动边缘计算下的安全 skyline 查询

Figure 1 (Color online) Secure skyline query in mobile edge computing

就返回结果 $R = \{P_3, P_7\}$, 如果 $E(Q_1)$ 是针对该街道和附近街道 (即数据集 $E(D_1)$ 和 $E(D_2)$), 通过与云服务器协作, 边缘服务器返回的结果 $R = \{P_7, P_8\}$, 其中 $E(\cdot)$ 表示加密算法. 在该模式下, 安全 skyline 查询一般涉及以下的隐私需求: (i) 数据的隐私; (ii) 查询请求的隐私; (iii) skyline 结果的隐私.

为解决移动边缘计算下基于位置数据的安全 skyline 查询问题, 目前面临以下两个研究挑战.

挑战一: 边缘服务上安全 skyline 查询的效率问题. 与云服务器上安全 skyline 查询不同, 边缘服务器因为受到计算、存储等资源限制, 其上执行的安全 skyline 查询协议应更加注重查询的轻量化, 即尽量减少耗时且频繁的操作, 如确定两个加密对象是否存在支配关系等. 因此, 如何设计新颖且统一的安全索引结构是实现轻量级安全 skyline 查询协议的关键.

挑战二: 云边协同中安全 skyline 查询的隐私计算问题. 在对多区域位置信息的安全索引进行 skyline 查询后, 云端需要对多区域的结果进行二次 skyline 计算, 同时要将结果发送给边缘侧进行再次合并计算. 因此, 在确保 skyline 结果隐私的前提下, 边缘侧 (云端) 如何对结果进行高效合并是边缘计算 skyline 查询处理的可扩展性难题.

综上所述, 本文提出的安全 skyline 查询方法主要包含 3 项贡献:

(1) 本文首次提出基于移动边缘计算的安全 skyline 查询系统模型, 并对安全模型给出详细的定义.

(2) 针对安全 skyline 查询的效率问题, 本文提出了轻量级的安全 R-tree 索引 (lightweight secure R-tree, SR*-tree), 其轻量化主要表现为: (i) 利用 Paillier 加密 [2] 和顺序公开加密 (order revealing encryption, ORE) [3] 方案对索引进行混合加密, 减少同态密文的计算开销; (ii) 对该加密结构进行必要的存储优化, 减少冗余的存储开销; (iii) 通过类似剪枝的方式避免对部分被支配元组的计算, 缩减了计算的数据规模. 同时, 本文为 SR*-tree 索引设计了半盲化结构, 实现了查询的不可链接性.

(3) 针对云边协同中的隐私计算问题, 本文首先基于 SR*-tree 索引提出了安全支配协议, 然后设计了面向位置信息的移动边缘计算 skyline 查询协议. 在保证隐私的前提下, 该安全查询协议不仅可以通过遍历 SR*-tree 索引来高效地返回精确的查询结果, 而且能对云和边的查询结果进行安全地 skyline 合并计算, 并将最终结果安全地返回给客户端.

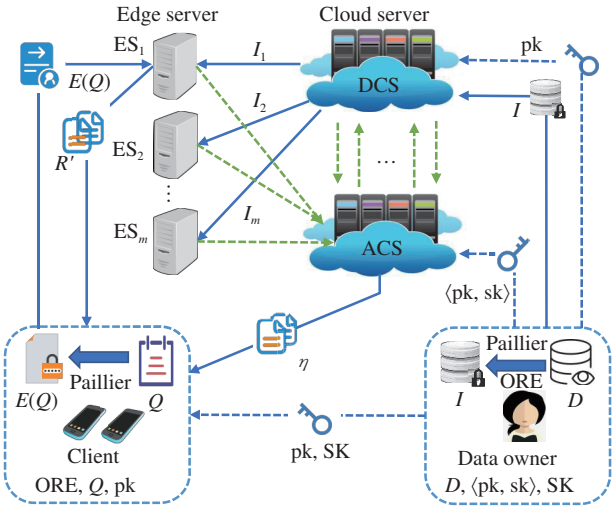

图 2 (网络版彩图) 系统模型

Figure 2 (Color online) System model

2 模型

系统模型. 在 MEC 下, 本文采用两个抗共谋的云服务器以及 m 个边缘服务器构建了面向位置信息的安全 skyline 查询架构. 如图 2 所示, 各实体功能介绍如下.

(1) 数据拥有者 (data owner, DO). DO 生成 Paillier 加密方案的密钥对 $\langle \mathrm{pk}, \mathrm{sk} \rangle$ 和 ORE 加密方案的密钥 SK, 并将明文数据集 D 按照区域划分并利用 Paillier 和 ORE 加密算法构建安全索引 $I = \{I_1, \ldots, I_i, \ldots, I_m\}$, 其中 I_i 表示边缘服务器 i (edge server i, ES_i) 所覆盖区域的数据点的安全索引. 接着, DO 将 I 上传至数据云服务器 (data cloud server, DCS), 同时将公钥 pk 通过安全信道发送给 DCS, 将 pk 和 SK 发送给客户端, 将 $\langle \mathrm{pk}, \mathrm{sk} \rangle$ 发送给辅助云服务器 (assistance cloud server, ACS).

(2) 客户端 (client, CT). 作为一个授权的客户端, 它向最近的边缘服务器提交被 Paillier 密码方案加密的查询 $E(Q)$. 在接收到边缘服务器返回的结果 R' 和辅助云服务器返回的结果 η 后, 它通过本地计算获得最终 skyline 结果.

(3) 边缘服务器 (ES). 当 ES_i 收到针对本区域数据的查询后, 它通过 ACS 的协助在安全索引 I_i 上执行安全 skyline 查询; 当收到的是针对多个区域的查询请求 (一般包含本区域的数据, 被称为多区域查询), ES_i 与 DCS 进行协作查询, 由 ES_i 和 ACS 分别返回 R' 和 η 给客户端.

(4) 云服务器 (cloud server). DCS 存储数据集 D 的安全索引 I, 并将索引按照区域分发给对应的 ES. 同时, DCS 还负责协助 ES 处理针对多区域的查询, 并将加密的结果返回给 ES. ACS 主要负责协助 ES (DCS) 执行安全 skyline 查询.

安全模型. 本文假设云服务器和边缘服务器都是 “半诚实” 的, 即它们会按照协议诚实地执行规定操作, 同时期望获取原始数据集和用户的查询 (位置信息), 但不会对数据和结果进行破坏或恶意篡改. 因此, 基于安全仿真模型 [4], 本文方案要保护以下 3 个方面隐私: (1) 云服务器和边缘服务器都无法获得关于原始数据集 D 的任何信息. (2) 任何关于查询 Q 的相关信息无法被任何参与方获悉. 同时, 作为查询的处理方 (云服务器或边缘服务器) 无法追踪查询的访问路径来获取额外信息, 即保护查询的不可链接性. (3) 除了对应客户端任何实体都无法获取查询结果 R. 换句话说, 其安全性要求本文协

议的仿真视图 $\mathrm{Sim}_{(\cdot)}^{\mathcal{A}}$ 在计算上与实际执行视图 $\mathrm{Real}_{(\cdot)}^{\mathcal{A}}$ 是不可区分的. 正式的定义如下:

定义1 给定安全参数 $\epsilon \in \mathbb{N}$, 当且仅当对于所有概率多项式时间的攻击者 $\mathcal{A}$, 存在一个有效的模拟器 $\mathcal{S}$ 使得对于输入加密查询 q 和加密数据集 $\mathcal{P}$ 下面等式是成立的, 则认为本文方案是安全的.

$$\left|\Pr[\mathrm{Real}_{(\cdot)}^{\mathcal{A}}(\epsilon, \mathcal{P}, q)] - \Pr[\mathrm{Sim}_{(\cdot)}^{\mathcal{A}}(\epsilon, \mathcal{P}, q)]\right| \leqslant \mathrm{negl}(\epsilon).$$

问题定义. 给定数据集 $D = \{P_1, \ldots, P_n\}$, 其中元组 $P_i \in D$ $(1 \leqslant i \leqslant n)$ 由空间位置属性 (被表示为 $P_i[1]$ 和 $P_i[2]$) 和非空间属性 (被表示为 $\langle P_i[3], \ldots, P_i[d]\rangle$) 组成.

定义2 (非空间支配) 给定 d 维空间上的任意两个元组 P_1 和 P_2, 当且仅当 $\forall\ 3 \leqslant i \leqslant d$, $P_1[i] \leqslant P_2[i]$, 那么 P_1 支配 P_2 (或 P_2 被 P_1 被支配), 被表示为 $P_1 \prec P_2$.

定义3 (支配) 给定位置查询元组 Q 和任意两个元组 P_1 和 P_2, 如果 (1) $P_1 \prec P_2$ 且 (2) P_1 比 P_2 更靠近查询 Q, 那么 P_1 关于查询 Q 支配 P_2, 被表示为 $P_1 \prec_Q P_2$.

定义4 (基于位置信息的安全 skyline 查询) 给定由数据集 D 所生成的安全索引 I 和加密查询 $E(Q)$, 协议所计算出的关于 $E(Q)$ 的 skyline 集合 R 满足 $\forall P_i, P_j \in R (i \neq j)$, $E(P_i) \not\prec_Q E(P_j)$ 且 $E(P_j) \not\prec_Q E(P_i)$, 同时满足对于 $\forall P_k \in (D - R)$, $\exists P_i \in R$, $E(P_i) \prec_Q E(P_k)$, 且满足安全性定义 1. 其中, skyline 结果的集合 R 也被表示为 $\mathrm{SKY}(I)$ 或 $\mathrm{SKY}(D)$.

3 轻量级安全 R − 树索引与支配协议

3.1 SR*-tree 索引

为了保护查询的不可链接性, 本文提出名为半盲化 R-tree 的结构. 在该结构中, 叶子结点由指向 D 中元组的数据对象构成. 普通非叶子结点由索引对象 (即 $(\mathrm{MBR}, \mathrm{pointer})$) 构成, 其中 pointer 是指向其孩子结点的指针, MBR 表示 d 维最小外接矩形, 即 $\mathrm{MBR} = (\mathrm{mbr}_1, \mathrm{mbr}_2, \ldots, \mathrm{mbr}_d)$. 其中, $\langle \mathrm{mbr}_1, \mathrm{mbr}_2\rangle$ 表示矩形左下角顶点的在空间维度上的值, $\langle \mathrm{mbr}_3, \ldots, \mathrm{mbr}_d\rangle$ 是指矩形左下角顶点在非空间维度的值. 位于半盲化 R-tree 的倒数第 2 层的结点 (也被称为盲化结点) 包含形式为 $(\mathrm{MBR}, \mathcal{B}(\cdot))$ 的条目, 其中 $\mathcal{B}(\cdot)$ 是计算其对应孩子结点的函数.

如图 3(a) 所示, 给定数据集 D 和一个查询点 Q, 其中 X 和 Y 为空间属性 (即 $P[1]$ 和 $P[2]$), 利用 R-tree 的数据分区方法对 D 进行分区, 其中结点容量 $c = 3$. 虽然对于高维数据而言, R-tree 很容易出现维度灾难, 但是正如 Kossmann 等 [5] 所认为, 大多数应用所处理的数据一般不超过五维. 因此, 对于 SR*-tree 这种特殊 R-tree 来说, 对其进行安全 skyline 查询仍然是有效的.

同时, 基于半盲化的 R-tree 和混合加密方式, 本文提出了 SR*-tree 索引来保护数据的隐私和安全. 如图 3(b) 所示, SR*-tree 的叶子结点包含数据对象的形式为

$$(\langle E(P[1]), E(P[2]), E(S_{\mathrm{NS}}(P))\rangle, \langle E(\mathrm{Enc}(P[3])), \ldots, E(\mathrm{Enc}(P[d]))\rangle), \tag{1}$$

其中 $E(\cdot)$ 表示 Paiillier 加密算法 [2], $\mathrm{Enc}(\cdot)$ 表示 ORE 加密算法 [3]. 通过预先计算 $E(S_{\mathrm{NS}}(P)) = E(\sum_{i=3}^{d} P[i])$, DCS 可以有效地减少计算开销, 并无需存储 $E(P[i])$ $(3 \leqslant i \leqslant d)$. 对于非叶子结点所包含的索引对象 e, 其数据形式为

$$(\langle E(\mathrm{mbr}_1), E(\mathrm{mbr}_2), E(S_{\mathrm{NS}}(\mathrm{MBR}))\rangle, \langle \mathrm{Enc}(\mathrm{mbr}_3), \ldots, \mathrm{Enc}(\mathrm{mbr}_d)\rangle, \mathrm{pointer}/\mathcal{B}(\cdot)), \tag{2}$$

同样, 通过计算 $E(S_{\mathrm{NS}}(\mathrm{MBR})) = E(\sum_{i=3}^{d} \mathrm{mbr}_i)$, 也可以优化效率和存储空间.

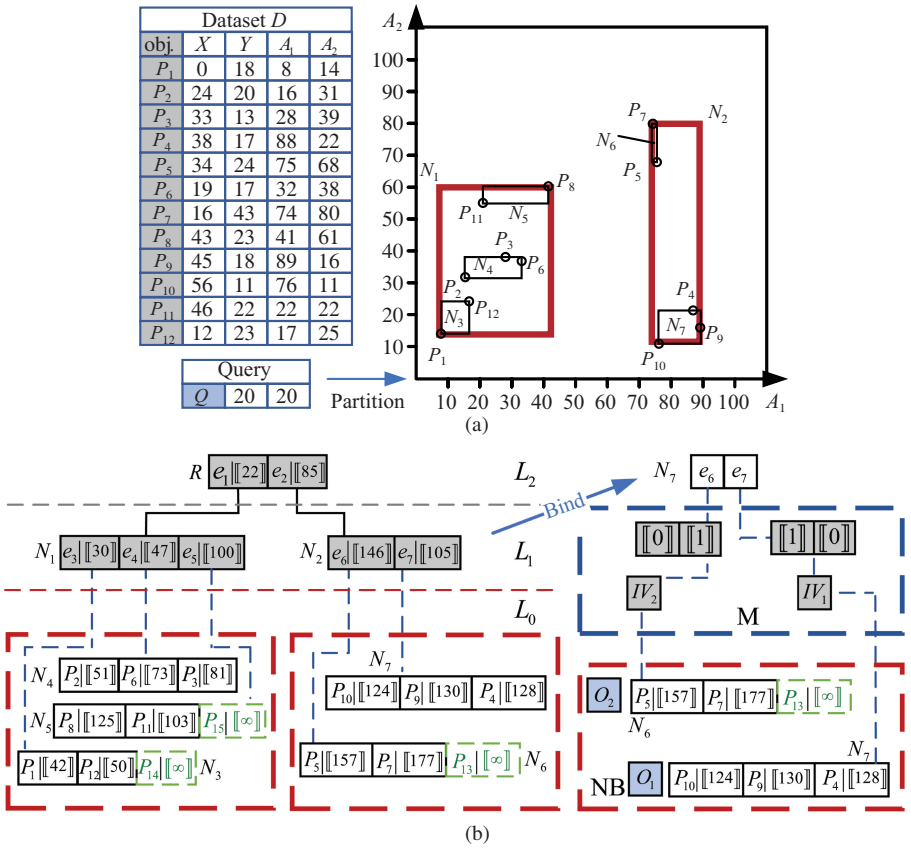

图 3 (网络版彩图) SR*-tree 索引

Figure 3 (Color online) SR*-tree index. (a) Tree partition of D; (b) SR*-tree with semi-blind structure, where $[\![\cdot]\!]$ indicates the ciphertext encrypted by Paillier. In the SR*-tree, the left side of vertical line is the object and the right side of it is the encrypted mindist(·)

在图 3(b) 中, 虚线表示该路径对程序是透明的, 这意味着云/边缘服务器无法根据访问模式来获得额外信息. 因此, 在半盲化结构中, 通过输入加密矩阵 M 和结点桶 NB 以计算函数 $\mathcal{B}(\cdot)$:

$$C_{ij} = \prod_{k=1}^{\theta} \mathrm{SM}(M_i[k], \mathrm{NB}_{kj}), \tag{3}$$

$$\mathcal{B}(M, \mathrm{NB}) = \begin{bmatrix} C_{11} & \cdots & C_{1c} \\ \vdots & C_{ij} & \vdots \\ C_{\theta 1} & \cdots & C_{\theta c} \end{bmatrix}, \tag{4}$$

其中, θ 表示父结点中索引对象的数量, $M = [\mathrm{IV}_1^{\mathrm{T}} \cdots \mathrm{IV}_\theta^{\mathrm{T}}]^{\mathrm{T}}$, $\mathrm{NB} = [E(\mathcal{O}_1)^{\mathrm{T}} \cdots E(\mathcal{O}_\theta)^{\mathrm{T}}]^{\mathrm{T}}$, 其中 $\mathrm{SM}(\cdot,\cdot)$ 表示基于 Paillier 的安全乘法 [6]. 为了保护查询的不可链接性, IV_i 由 $E(1)$ 和 $E(0)$ 组成, 它们通过 SM 协议来定位叶子结点, $E(O_i)$ 是 IV_i 所对应的被加密的叶子结点. $\mathcal{B}(\cdot)$ 的计算结果是一个矩阵, 其中每行代表的是一个叶子结点, 并且其顺序与它的父结点索引对象一一对应. 如图 3(b) 所示, 它是一个典型 $c = 3$ 且 $\theta = 2$ 的半盲化结构.

为了抵抗基于叶子结点容量的推理攻击 (即, 由于某些叶子结点没有完全填充, 云/边缘服务器可以通过观察结点的容量来确定两个查询是否访问了相同的结点), 数据所有者将生成合适的噪声元组

添加到缺少足够数据点的结点中. 如图 3(b) 所示, 虚线的条目是被添加的噪声元组. 由于填充的数据点 (它们的值设置为足够大) 将会被某一个 skyline 点支配, 因此它们不会影响查询结果的准确性. 值得注意的是, 数据集 D 的 SR*-tree 索引通常由数据所有者维护. 虽然 SR*-tree 中的半盲结构会带来额外的计算开销, 但是基于 SR*-tree 的 skyline 查询仅需要在部分数据点上执行支配运算, 而无需遍历整个数据集. 同时, 被触发的半盲结构的数量仅与 skyline 结果中数据点的数量相关, 因此该结构对于大多数结果稀疏但数据量大的应用来说是非常有利的.

3.2 基于 SR*-tree 的安全支配协议

基于 SR*-tree 的安全支配 (secure dominance based on SR*-tree, STDM) 协议的目的是计算加密对象 e 和加密 skyline 点 s 之间的支配关系. 假设 DCS 拥有来自 R 中的 skyline 点 s、来自安全索引 I 的对象 e、s 到 $E(Q)$ 的 Euclid 距离 d_s, 以及从 e 到 $E(Q)$ 的 Euclid 距离 d_e, ACS 拥有私钥/公钥对 $\langle \mathrm{sk}, \mathrm{pk}\rangle$, 其中关于 e.MBR (或 $e.P$) 和 $s.P$ 的内容不会被泄露给服务器 DCS 和 ACS.

DCS 通过式 (5) 将 d_s 和 d_e (已添加噪声 r, 见 4.1 小节) 打包到 β_1, 值得注意的是 $||d_s||, ||d_e|| < ||N||/2$, $||\cdot||$ 表示为 "·" 的比特长度.

$$\beta_1 = \prod_{i=1}^{2} d_i^{\,2^{\bar{\sigma}(2-i)}}, \tag{5}$$

其中 $\bar{\sigma} = ||N||/2$. 随后, DCS 将 β_1 发送给 ACS. 在接收到值后, ACS 将其解密为 β_1', 并将其解包为 $\langle d_1, d_2\rangle$, 具体计算如下:

$$d_i = \left\lfloor \frac{\beta_i'}{2^{\bar{\sigma}(2-i)}} \right\rfloor, \tag{6}$$

其中 $\beta_{i+1}' = \beta_i' - d_i \times 2^{\bar{\sigma}(2-i)}$. 通过比较 d_1 与 d_2 获得 f, 并将 f 发送给 DCS. 如果 $f = 1$ (即 $d_1 < d_2$), 则表示 s 距离 Q 更近; 反之, e 距离 Q 更加近. 如果 e 是数据对象, 那么需要 ACS 协助解密 $\langle E(\mathrm{Enc}(e.P[3])), \ldots, E(\mathrm{Enc}(e.P[d]))\rangle$, 以获得 $\langle \mathrm{Enc}(e.P[3]), \ldots, \mathrm{Enc}(e.P[d])\rangle$. DCS 通过比较其和 $\langle \mathrm{Enc}(s.P[3]), \ldots, \mathrm{Enc}(s.P[d])\rangle$ 来获得非空间属性的数值关系 u. 同理, 如果 e 是索引对象, DCS 可以直接比较 e 和 s 来获得 u. 最后, DCS 可计算得到支配关系 $\Phi = f \wedge u_1 \cdots \wedge u_{d-2}$.

4 面向位置信息的移动边缘计算安全 skyline 查询协议

4.1 面向单区域位置信息的安全 skyline 查询协议

预处理. 数据拥有者基于 SR*-tree 生成安全索引 I, 并将其上传至云服务器 DCS, 并由 DCS 将 I_i 发送给对应的边缘服务器 ES. 当客户端向 ES 发送查询 $E(Q)$ 时, ES 需要安全地计算 $E(Q)$ 到对象 e (来自 I_i) 的加密距离 $E(\mathrm{dist}(e, Q))$. 当 e 是数据对象时, ES 基于 SSED 算法 [7] 计算平方 Euclid 距离 $E(\mathrm{dist}^2)$. 当 e 是索引对象时, ES 根据 e 相对于 Q 的位置来计算平方 Euclid 距离 $E(\mathrm{dist}^2)$. 具体如下: 如果 Q 所在的空间维度 ($\langle Q[1], Q[2]\rangle$) 均不位于 e.MBR 的空间维度范围, 那么 $E(\mathrm{dist}^2(e, \mathcal{Q})) = \mathrm{SSED}(e, E(\mathcal{Q}))$. 如果 Q 仅某一个维度 $Q[1](Q[2])$ 位于 e.MBR 的空间维度范围, 那么平方 Euclid 距离 $E(\mathrm{dist}^2(e, \mathcal{Q})) = E(|e.\mathrm{mbr}_2 - \mathcal{Q}[2]|^2)(E(\mathrm{dist}^2(e, \mathcal{Q})) = E(|e.\mathrm{mbr}_1 - \mathcal{Q}[1]|^2))$. 接着, ES 计算 $\widetilde{\gamma} = E(\mathrm{dist}^2(e, \mathcal{Q}))^{r^2}$ 并将其发送到 ACS, 其中 r 是噪声数据 (每轮查询 r 保持不变). 收到 $\widetilde{\gamma}$ 后, ACS 将其解密并获得 $\widehat{\gamma}$. 然后, ACS 将 $E(\sqrt{\widehat{\gamma}})$ 返回给 ES. 之后, ES 令 $E(\mathrm{dist}(e, \mathcal{Q})) = E(\sqrt{\widehat{\gamma}})$. 值得注意的是, $E(\mathrm{dist}(e, \mathcal{Q}))$ 的明文值是实际距离的 r 倍, 且被存储用于计算支配关系.

安全 skyline 查询. 其主要思想源自以下两个引理.

引理1 给定查询点 Q, 如果对象 e 支配了另一数据对象 e', 可知 $\text{mindist}(e,Q) < \text{mindist}(e',Q)$; 如果对象 e 支配了索引对象 N, 可知 $\text{mindist}(e,Q) < \text{mindist}(N,Q)$.

引理2 通过 mindist 升序遍历 SR*-tree (一种特殊的 R-tree), 其每一个被加入候选 skyline 集合中的数据对象即是最终的 skyline 点.

引理 2 表明如果以升序遍历 SR*-tree 索引 I_i, 则只需要进行一次扫描即可发现 skyline 结果, 其中 mindist 表示对象到查询点 Q 的最小距离. 基于此, 本文提出了基于安全索引的 skyline 查询算法 (secure-index-based skyline queries, SISQ) (见算法 1). 首先, ES 初始化一个 skyline 结果集 $R = \emptyset$ 和一个最小堆 $\mathcal{H}$, 该最小堆保存来自 SR*-tree 索引 I_i 的对象, 利用安全最小 (secure minimum, SMIN) 算法 [6] 将最小 $\text{mindtist}_E(\cdot)$ 的 e 置于 $\mathcal{H}$ 的顶部. 当 e 是数据对象时, ES 计算 $\text{mindist}_E(e,\mathcal{Q}) = E(\text{dist}(e,\mathcal{Q})) \times E(S_{\text{NS}}(P))^r$. 当 e 是索引对象时, ES 计算 $\text{mindist}_E(e,\mathcal{Q}) = E(\text{dist}(e,\mathcal{Q})) \times E(S_{\text{NS}}(\text{MBR}))^r$. 值得注意的是, 该 $\text{mindist}_E(\cdot)$ 的明文值也是实际值的 r 倍.

Algorithm 1 Secure-index-based skyline queries algorithm

Input: ES has the root of I_i sRoot and ACS has sk.
Output: ES $\leftarrow$ skyline set R.

```
1: R = ∅, insert sRoot into a min-heap H ordered by mindist_E(·);
2: while H is not empty do
3:    Get the top element e from H;
4:    if e is an index object then
5:       if e is dominated by some object in R through STDM algorithm then
6:          Prune e;
7:       else
8:          if e is a blind entry then
9:             {e_1, ..., e_i, ...} = B(E(e));
10:         end if
11:         for each child e_i of e do
12:            if e_i is dominated by some object in R through STDM algorithm then
13:               Prune e_i;
14:            else
15:               Insert e_i into H;
16:            end if
17:         end for
18:      end if
19:   end if
20:   if e is a data object then
21:      if e is not dominated by each object in R through STDM algorithm then
22:         R ← R ∪ (⟨E(P[1]), E(P[2])⟩, mindist_E, ⟨Enc(P[3]), ..., Enc(P[d])⟩);
23:      end if
24:   end if
25: end while
```

算法具体如下, ES 将 SR*-tree 索引 I_i 的根结点对象 sRoot (包含 $\text{mindist}_E(\cdot)$) 插入到 $\mathcal{H}$ 中, 并从 $\mathcal{H}$ 顶部取出对象 e. 如果 e 是索引对象, ES 通过 STDM 协议来确定 e 是否由 R 中的 skyline 点支配, 如果 e 是被支配的对象, 则 e 会被剪枝. 否则, 如果 e 是未被支配的对像, 同时也是盲化条目, ES 需要通过函数 $\mathcal{B}(\cdot)$ 计算其孩子对象 e_i. 如果不是盲化条目, 可以直接通过指针获得其孩子结点 e_i. 对于 e 的每个孩子对象 e_i, 仅当它不能被 R 中的任意 skyline 点支配时, 才会被插入 $\mathcal{H}$ 中. 否则, 该孩子对象

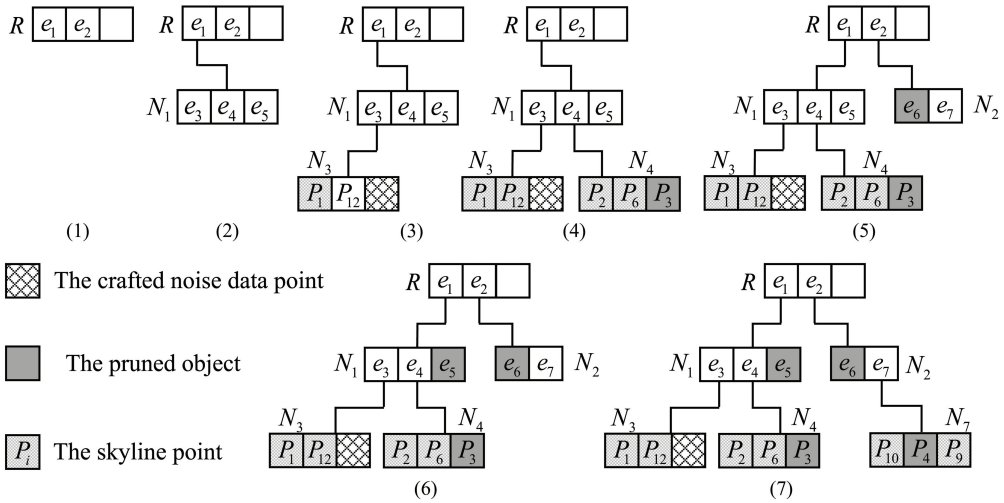

图 4 SR*-tree 的遍历

Figure 4 Sample of the traversal of SR*-tree

e_i 也会被剪枝. 这样可以减少 $\mathcal{H}$ 中对象的数量, 减少 SMIN 算法的时间开销. 如果 e 是数据对象且未被 R 中的任意 skyline 点支配, 则将其对应的 $(\langle E(P[1]), E(P[2])\rangle, \text{mindist}_E, \langle \text{Enc}(P[3]), \ldots, \text{Enc}(P[d])\rangle)$ 插入 R 中. 算法继续执行直到 $\mathcal{H} = \emptyset$ 停止.

图 4 显示的是安全 skyline 查询算法遍历 SR*-tree 的过程, 其中 SR*-tree 与图 3(b) 中的相同. 具体来说, ES 从根结点遍历 SR*-tree, 并将其加密对象 (e_1, e_2) 和对应的 $\text{mindist}_E(\cdot)$ 加入最小堆 $\mathcal{H}$ 中. 接着, ES 从 $\mathcal{H}$ 中取出具有最小明文值的 $\text{mindist}_E(\cdot)$ 的对象 e_1, 并向 $\mathcal{H}$ 中插入其孩子对象 (e_3, e_4, e_5) 及对应的 $\text{mindist}_E(\cdot)$. 下一个具有最小明文值的 $\text{mindist}_E(\cdot)$ 的对象是 e_3, 其孩子对象 (P_1, P_{12}, P_{14}) 可以通过函数 $\mathcal{B}(\cdot)$ 计算得到, 并插入到 $\mathcal{H}$ 中. 接下来从 $\mathcal{H}$ 中取出的对象是 P_1, 由于其无法被支配, 所以被添加到 R 中. 接着, 从 $\mathcal{H}$ 中取出 e_4 进行扩展, 由于 (P_2, P_6, P_3) 均无法被 P_1 支配, 所以都被添加进 $\mathcal{H}$ 中. 然后, 依次从 $\mathcal{H}$ 中取出 P_{12}, P_2, P_6 和 P_3, 其中除 P_3 均是 skyline 点. 接下来, 从 $\mathcal{H}$ 中取出 e_2 进行扩展, 由于其孩子对象 e_6 被 P_2 支配, 所以被剪枝. 它的另一个孩子对象 e_7 被添加进 $\mathcal{H}$ 中. 当取出 e_5 后, 由于其能被 P_1 支配, 所以被剪枝. 算法一直执行直到 $\mathcal{H} = \emptyset$ 停止, skyline 点包括 P_1, P_{12}, P_2, P_6, P_{10} 和 P_9. 值得注意的是, 图 4 的对象 e_i 均是加密对象, 且 N_3, N_4 和 N_7 均由 $\mathcal{B}(\cdot)$ 计算而得.

返回结果. 一旦 ES 获得加密的 skyline 结果 R $(E(s) \in R)$ 后, 它会通过计算 $\eta'[j] = E(s[j] + r[j]) = E_{\text{sk}}(s[j]) \times E_{\text{sk}}(r[j])$ (其中, $1 \leqslant j \leqslant 2$) 将随机噪声 $r[j]$ 添加到每一个 $s[j]$, 然后将噪声 r 和 $\langle \text{Enc}(P[3]), \ldots, \text{Enc}(P[d])\rangle$ 发送给客户端, 将 η' 发送给 ACS. 接着, ACS 通过计算 $\eta[j] = D_{\text{sk}}(\eta'[j])$ 解密 $\eta'[j]$, 并将 $\eta[j]$ 发送给客户端. 最终, 客户端通过计算 $s[j] = \eta[j] - r[j]$ 获得 skyline 结果的空间属性部分, 通过 SK 解密获得非空间属性部分.

4.2 面向多区域位置信息的安全 skyline 查询协议

当客户端查询请求 $E(Q)$ 是针对多个区域的数据时 (包含查询所在位置的区域), 边缘服务器 ES 除了针对 $E(Q)$ 对缓存中的安全索引进行安全 skyline 查询, 而且会将 $E(Q)$ 转发给云服务器 DCS, 由 DCS 针对其他区域的安全索引分别进行安全 skyline 查询, 并对多个输出进行 skyline 合并计算. 最终, DCS 会将查询结果返回给 ES, 由 ES 进行 skyline 合并计算并返回给客户端. 由于针对数据的预处理

和基于安全索引的 skyline 查询与 4.1 小节类似, 所以本小节不做过多赘述. 从该协议步骤来看, 如何对安全 skyline 查询的结果进行有效地合并是面向多区域的位置信息安全 skyline 查询协议的关键所在. 因此, 本节着重阐述安全 skyline 合并 (secure skyline mergence, SSM) 算法.

首先, 本文引入 skyline 的可加性定义 (见定义 5). 根据该原理, 云服务器 DCS (边缘服务器 ES) 只需要对所有 SISQ 算法的输出进行一次安全 skyline 计算, 该计算也被称为安全 skyline 合并.

定义5 给定数据集 $D = D_1 \cup \cdots \cup D_i \cup \cdots \cup D_m$, 其中 D_i 是第 i 个子数据集, 那么 D 的 skyline 结果为 $\mathrm{SKY}(D) = \mathrm{SKY}(\mathrm{SKY}(D_1) \cup \cdots \cup \mathrm{SKY}(D_m))$.

以云服务器 DCS 执行 SSM 算法为例, DCS 先将各子数据集的 skyline 结果并合并为新数据集 D', 并通过 SMIN 算法获得最小的 mindist$(\cdot)$ 的加密值 α, 然后利用安全整数比较算法 [8] 比较 α 与 D' 中每一元组的 mindist$_E(\cdot)$ 并获得加密的比较结果 φ. DCS 将其发送给 ACS 解密并把 $D_{\mathrm{sk}}(\varphi_i) = 1$ (即对应元组的 mindist$(\cdot)$ 与 α 的明文相等) 的编号返回给 DCS, DCS 将对应的元组 $E(P_{\min})$ 添加到 $\widehat{R}$ 中. 如果出现多个 $D_{\mathrm{sk}}(\varphi_i) = 1$ 的情况, ACS 一般会取第 1 个元组的编号返回给 DCS. 接着, DCS 利用 STDM 算法去判断 D' 中其他元组与 $P_{\min}$ 的支配关系, 删除其中被支配的元组和 $P_{\min}$. 接着, 算法继续执行直到 $D' = \emptyset$ 停止.

当服务器 DCS 返回查询结果 $\widehat{R}$ 给 ES 后, ES 针对 $\widehat{R}$ 与边缘安全索引的查询结果 $\bar{R}$ 执行 SSM 算法以获得 skyline 最终结果 R. 最后, ES 利用与 4.1 小节相同的方式将结果 R 返回给客户端.

5 复杂度与安全性分析

复杂度分析. 在预处理阶段, 算法需要执行 1 次 SSED 协议, 所以要进行 $4+\lceil 2/\lambda \rceil$ 次加密和 $\lceil 2/\lambda \rceil$ 次解密. 因此, 要计算 $E(\mathrm{dist}(e, Q))$, 需要执行 $5+\lceil 2/\lambda \rceil$ 次加密和 $1+\lceil 2/\lambda \rceil$ 次解密. 本文假设添加到 $\mathcal{H}$ 的对象数量为 l, 其中索引对象的数量为 l_1, 数据对象的数量为 l_2, 并且 $l_1 + l_2 = l$. 由于 SISQ 算法需要预处理 l 个对象, 所以需要 $5l + \lceil 2l/\lambda \rceil$ 次加密和 $l + \lceil 2l/\lambda \rceil$ 次解密. SISQ 算法在第 5 和 10 行都调用了 STDM 算法, 其中存在对同一对象重复调用的情况. 因此, 它需要略多于 $dl_2 + l_1 - l_2$ 次解密. 假设 skyline 点的数量为 λ_s, 可知 $\lambda_s < l_2 < l \ll n$, 所以 SISQ 算法的复杂度约为 $\mathcal{O}(d\lambda_s)$. 由于 SSM 算法的复杂度也与候选 skyline 点个数 λ'_s ($\lambda_s < \lambda'_s$) 相关, 所以面向多区域位置信息的安全 skyline 查询算法的复杂度也约为 $\mathcal{O}(d\lambda_s)$.

安全性分析. 本文采用安全仿真模型 [4] 来证明所提出的算法的安全性, 以 STDM 算法为例.

定理1 对于半诚实的攻击者 $\mathcal{A}$ 而言, 如果存在模拟器 $\mathcal{S}$ 使得概率 $\Pr(\mathrm{Real}^{\mathcal{A}}_{\mathrm{STDM}}) - \Pr(\mathrm{Sim}^{\mathcal{A}}_{\mathrm{STDM}})$ 是可以忽略的, 那么 STDM 算法是安全的.

证明 为了证明该算法的安全性, 本文构造如下的真实视图 $\mathrm{Real}^{\mathcal{A}}_{\mathrm{STDM}}$ 和模拟视图 $\mathrm{Sim}^{\mathcal{A}}_{\mathrm{STDM}}$.

$\mathrm{Real}^{\mathcal{A}}_{\mathrm{STDM}}$: 将 STDM 协议拆成以下两个部分执行, 给定输入 d_e 和 d_s, DCS 根据协议将已添加噪声的 d_e 和 d_s 打包成 β_1 发送给 ACS. ACS 解密 (包含解包) 它们并进行比较, 通过实验输出结果 f ($f \in \{0, 1\}$); 给定输入 s 和 e, 然后获取由 ORE 算法加密的 s 和 e 的非空间属性, 并通过实验输出非空间属性的数值关系 u ($u_i \in \{0, 1\}$).

$\mathrm{Sim}^{\mathcal{A}}_{\mathrm{STDM}}$: 模拟器 $\mathcal{S}$ 收到 f 和 u, 然后生成随机数 $\widetilde{d_s} = [\![r_1]\!] \times [\![r_2]\!]^{1-f}$ 和 $\widetilde{d_e} = [\![r_1]\!] \times [\![r_2]\!]^{f}$, 其中 $[\![\cdot]\!]$ 表示 Paillier 加密算法, r_1 和 r_2 满足 $r_1 \neq r_2 \neq 0$ 且 $||(r_1 + r_2)|| < ||N||/2$. 同理, 可以根据 u 和 e 的类型构造出 $\widetilde{e}$ 和 $\widetilde{s}$ 的非空间属性. 之后, $\mathcal{S}$ 形成模拟数据 $\widetilde{d_s}$, $\widetilde{d_e}$, $\widetilde{e}$ 和 $\widetilde{s}$, 并将它们作为 STDM 算法的输入, 通过实验输出结果.

基于模拟器 $\mathcal{S}$, 在概率多项式时间 (probabilistic polynomial time, PPT) 时间内, 敌手无法区分 $\mathrm{Sim}^{\mathcal{A}}_{\mathrm{STDM}}$ 和 $\mathrm{Real}^{\mathcal{A}}_{\mathrm{STDM}}$, 因为它们的输出是相同的. 也就是说, 模拟视图在计算上与实际执行视图是没有区别的. 此外, 对于特定的隐私, paillier 密码系统的语义安全性、ORE 密码算法的安全性 (攻击者不能在概率多项式时间内解密数据) 和混入的噪声保证了数据和查询的隐私性. 综上所述, STDM 算法是安全的.

同理, SISQ 和 SSM 算法也可以在半诚实的安全模型下被证明是安全的.

定理2 (组合定理, composition theorem [4]) 给定一个由多个子协议组成的协议 υ, 如果所有的子协议是安全的且它们的中间结果是随机或伪随机的, 那么该协议 υ 是安全的.

接着, 面向位置信息的移动边缘计算安全 skyline 查询协议的安全性可以被下面的定理所保证.

定理3 对于半诚实的攻击者 $\mathcal{A}$, 面向位置信息的移动边缘计算安全 skyline 查询协议是安全的.

证明 基于定理 1, 可知本协议中所涉及的子协议是安全的. 同时, 根据定理 2, 可以很容易证明本协议是安全的.

6 实验评估

6.1 实验设置

仿真实验以 java 为主要编程语言, 并运行在 Windows 10 操作系统的物理主机上 (硬件参数为 Intel Core I5-8400, 2.8 GH, 8 GB). 在实验中, 边缘服务器 ES、云服务器 DCS 和 ACS 运行在同一台机器上, 并且使用线程来模拟它们. 实验所采用的 Paillier 和 ORE 加密算法的密钥长度都为 512 位. 同时, 实验生成与文献 [9] 类似的相关 (CORR)、独立 (INDE) 和反相关 (ANTI) 数据集, 并根据加利福尼亚街段质心的真实数据1)生成数据集 LOCA, 其非空间属性利用 0∼100000 的均匀分布生成.

需要注意的是, 在评估 SR*-tree 构造时, 维度的数量 d 仅包含非空间属性; 在评估算法查询效率时, 维度的数量 d 包含非空间属性和一个距离属性 $\mathrm{dist}(\cdot,\cdot)$.

6.2 SR*-tree 的构造

实验评估了 SR*-tree 的内存存储开销. 由于其存储开销与数据类型无关, 因此实验选择了 ANTI 数据集来评估元组数量 n、维度 d 和结点容量 c 对存储开销的影响. 如图 5(a) 所示, SR*-tree 的储存开销随着 n 的增加而线性增加. 当维度 d 增加时, 其存储开销也呈现线性增加. 与其他参数不同, 随着结点容量 c 的增加, SR*-tree 的储存开销呈亚线性下降的趋势 (图 5(b)). 这是因为随着 c 的继续增加, SR*-tree 的高度和结点数量将减少, 所以 SR*-tree 所占的储存空间就会减少.

与存储开销类似, 实验基于 ANTI 数据集评估了 SR*-tree 的构造时间. 如图 6(a), 其构造所需时间随着 n 的增加而线性增加. 在图 6(a) 中, 可以发现 SR*-tree 的构建时间与元组的维度 d 成正比. 与之相反, 当结点容量 c 增加时, SR*-tree 的构建时间呈亚线性下降的趋势 (图 6(b)).

6.3 面向单区域位置信息的安全 skyline 查询算法的效率

实验通过分别更改参数 n, d 和 c, 同时与基础安全 skyline 协议 (basic secure skyline protocol, BSSP) [10, 11] 对比, 评估了 SISQ 算法的效率, 即在 ES 和 ACS 上的计算时间 (包含预处理时间).

1) http://www.rtreeportal.org/.

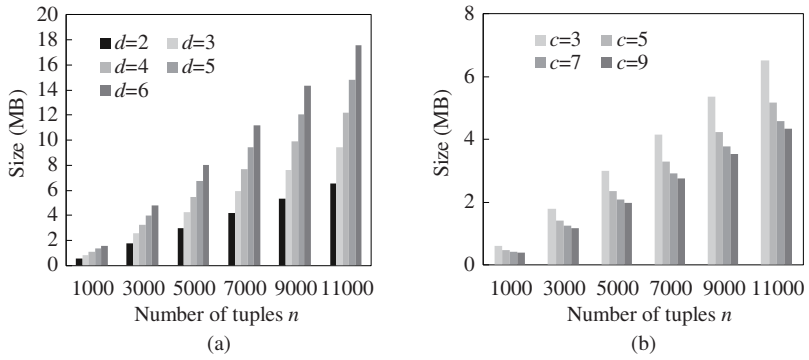

图 5 SR*-tree 的内存存储开销

Figure 5 Memory cost of SR*-tree. (a) Varying d and n ($c = 3$); (b) varying n and c ($d = 2$)

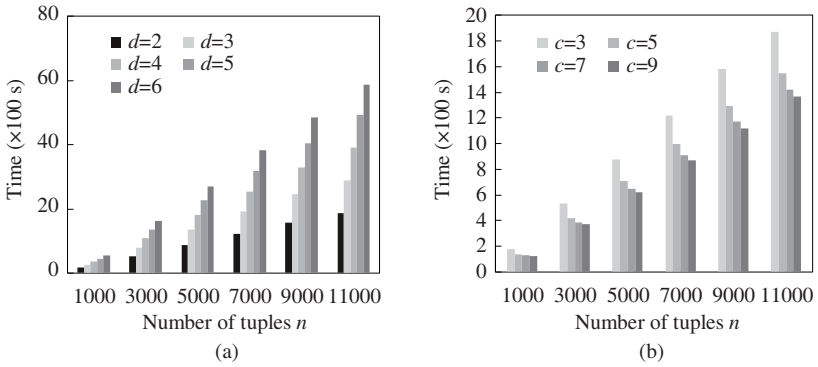

图 6 构建 SR*-tree 的时间开销

Figure 6 Construction time of SR*-tree. (a) Varying d and n ($c = 3$); (b) varying n and c ($d = 2$)

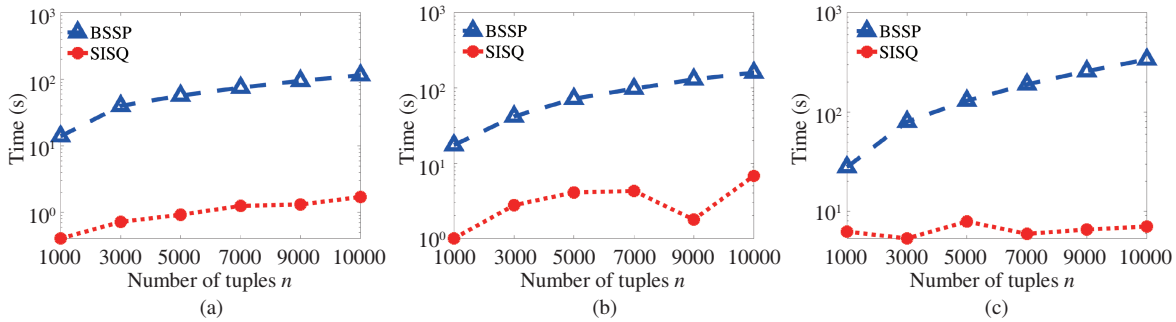

图 7 (网络版彩图) 参数 n 的影响 ($d = 2, c = 3$)

Figure 7 (Color online) The effect of n. (a) Time overhead of CORR; (b) time overhead of INDE; (c) time overhead of ANTI

图 7 展示了 n 对计算时间成本的影响. 在安全性模型基本一致的情况下, SISQ 在这 3 个数据集上的查询效率明显更高. 之所以有明显的效率提升是因为 SISQ 算法使用本文所提出的 SR*-tree 索引, 通过遍历 SR*-tree 可以大幅度减少对部分被支配数据元组的计算开销. 同时, 从 SISQ 算法在这 3 个数据集上的执行时间来看, 其时间成本与 n 几乎没有相关性, 这与前文的复杂度分析基本一致, 也进

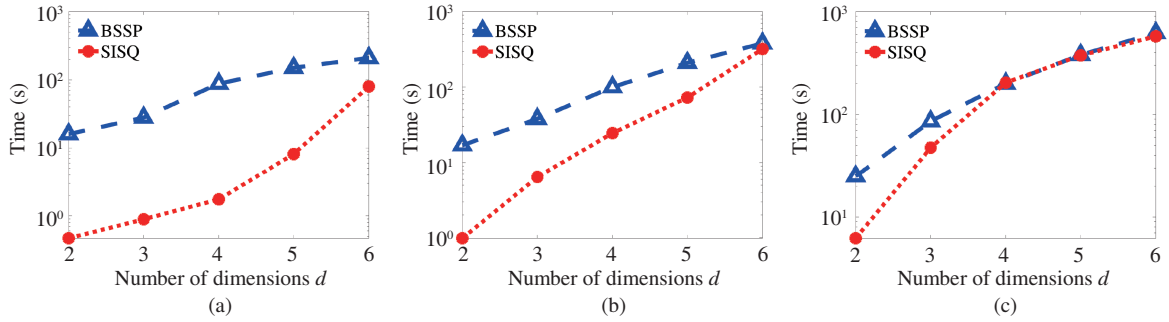

图 8　(网络版彩图) 参数 d 的影响 ($n = 1000, c = 3$)

Figure 8　(Color online) The effect of d. (a) Time overhead of CORR; (b) time overhead of INDE; (c) time overhead of ANTI

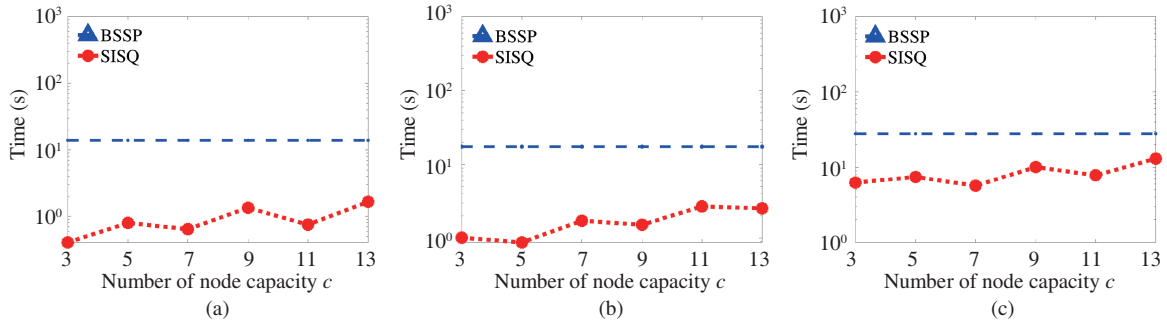

图 9　(网络版彩图) 参数 c 的影响 ($n = 1000, d = 2$)

Figure 9　(Color online) The effect of c. (a) Time overhead of CORR; (b) time overhead of INDE; (c) time overhead of ANTI

一步表明该算法对 n 是不敏感的.

图 8 展示了 d 对计算时间成本的影响. 从图中可观察到由于引入了 SR*-tree 索引, SISQ 算法比 BSSP 算法效率更高. 但是, 如图 8(c) 所示, 当维数 $d \geqslant 4$ 时, SISQ 算法的计算时间成本已呈现出接近 FSSP 的趋势, 这表明其对 d 是敏感的. 幸运的是, 大多数应用程序涉及到的维度不超过五维 [5], 因此本文的协议仍具有广泛适用性.

图 9 展示了 c 对计算时间成本的影响. 根据观察, 在此 3 个数据集上, 计算时间成本随结点容量 c 的增加而增加. 这是因为随着结点容量的增加, SR*-tree 对被支配结点的修剪效果变得更糟糕. 同样由于修剪效果的不确定性等因素, 计算时间成本表现出锯齿状波动. 如图 9 所示, 水平直线表示 BSSP 算法在 $n = 1000$, $d = 2$ 的情况下的计算时间成本 (BSSP 算法没有参数 c), 实验可观察到 SISQ 算法的执行时间一直位于其之下.

6.4　面向多区域位置信息的安全 skyline 查询算法的效率

实验通过改变子数据集的数量 κ ($n = 1000$, 维度 $d = 3$, $c = 3$) 评估了云边协同的安全 skyline 查询的效率, 即算法在 ES, DCS 和 ACS 上的总计算时间, 其中单个子数据集的安全索引位于 ES 处, 其余 $\kappa - 1$ 个子数据集的安全索引位于 DCS 处. 如图 10 所示, 随着数据集数量的增加, 计算的时间开销也随之增加. 但是, 当 $\kappa = 6$ 时, 针对数据集 INDE 的查询时间出现减少. 针对该情况, 实验统计并分析了算法各阶段所产生的候选 skyline 点数量 (如表 1 所示), 当 $\kappa = 6$ 时, DCS 所计算出的候选

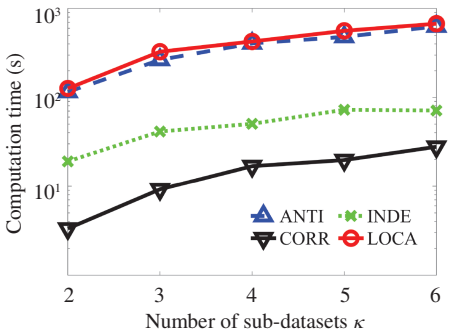

图 10 (网络版彩图) 面向多区域位置信息的计算时间

Figure 10 (Color online) Calculation time for multiple location-based data

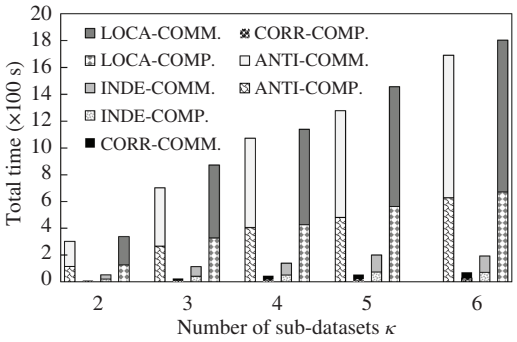

图 11 面向多区域位置信的计算和通信总时间

Figure 11 Total calculation and communication time for multiple location-based data

表 1 Skyline 点的个数

Table 1 Number of skyline points. The number of (candidate) skyline points is masked on the right side of the colon. Moreover, [·] indicates the number of skyline points in the corresponding sub-datasets and the number of merged skyline points is marked on the right side of the arrow

		$\kappa = 2$	$\kappa = 4$	$\kappa = 5$	$\kappa = 6$
ANTI, edge: 63	Cloud:	$[85] \to 85$	$[85, 85, 89] \to 122$	$[85, 85, 89, 74] \to 121$	$[85, \ldots, 74, 99] \to 136$
	R:	78	118	123	139
CORR, edge: 9	Cloud:	$[11] \to 11$	$[11, 14, 22] \to 21$	$[11, 14, 22, 9] \to 24$	$[11, \ldots, 9, 17] \to 33$
	R:	15	22	24	32
INDE, edge: 29	Cloud:	$[34] \to 34$	$[34, 31, 32] \to 45$	$[34, 31, 32, 35] \to 57$	$[34, \ldots, 35, 18] \to 46$
	R:	46	50	57	51

skyline 点的数量较 $\kappa = 5$ 时有所减少, 这使得 ES 所合并的数据规模也出现减少, 因此会出现计算时间减少的情况. 很显然, SSM 算法的时间成本与待合并的候选 skyline 点的数量是正相关的. 同时, 实验还发现 SISQ 算法的时间开销与候选 skyline 点的数量也是正相关的, 这与复杂度分析基本一致.

为了反映实际的通信开销, 将 ES, DCS 和 ACS 置于不同的物理机器上运行. 图 11 展示了云边协同下安全 skyline 查询的计算和通信时间成本. 据观察, 在这 4 种类型的数据集上, 其通信时间成本占总时间的一半以上.

为了减少安全 skyline 计算的时间开销, 本文在云服务器上对算法进行并行化实现. 针对各个子数据集 (安全索引 I_i) 进行并行化操作, 由每一个空闲线程分别去执行 SISQ 算法, 其结果再由其他空闲线程利用 SSM 算法进行合并操作. 实验对 6 个子数据集 (安全索引 $\{I_1, \ldots, I_6\}$) 进行云边协同安全 skyline 查询, 其中安全索引 I_1 位于 ES 上, 其余安全索引位于 DCS 上. 在每一个安全索引中, $n = 1000$, 维度为 $d = 3$, 结点容量 $c = 3$. 如图 12(a) 所示, 随着线程数量的增加, 其计算时间出现明显减少. 同时, 由于 SSM 算法也会占据一定的时间开销, 所以实验将候选 skyline 点集合分为多个数据块, 将每个数据块分配给一个线程以计算 SKY(·). 当接收到来自两个线程的 skyline 结果后, 主线程将它们合并为一个新的数据块, 该数据块又被分配给另一空闲线程以计算 SKY(·). 以此继续, 直到获得 skyline 结果. 如图 12(b) 所示, 随着线程的增加, 其计算时间会进一步降低. 但对于 INDE 和 CORR 数据集而言, 其计算时间下降的幅度有限, 这主要是因为他们执行 SSM 算法所消耗的时间占比较少 (待合并的候选 skyline 点较少).

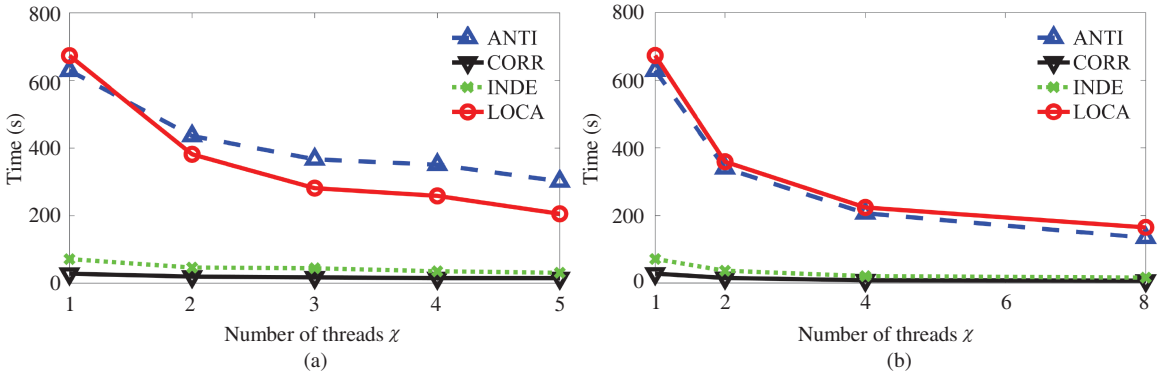

图 12　(网络版彩图) 多线程化实现

Figure 12　(Color online) Muti-threading implementation. (a) Multi-threading implementation based on subdatasets; (b) multi-threading implementation based on subdatasets and candidate skyline sets

7　相关工作

本文主要研究移动边缘计算场景下面向位置信息的安全 skyline 计算问题. 因此, 主要从 skyline 计算处理、集中式安全 skyline 计算协议和分布式安全 skyline 计算协议 3 个方面进行介绍.

Börzsönyi 等 [9] 首先提出块嵌套循环 (block nested loop, BNL) 的 skyline 查询算法. 随后, Kossmann 等 [5] 使用最近邻居 (nearest neighbor, NN) 来计算 skyline 查询的在线算法. Papadias 等 [12] 通过分析 NN 方法在 I/O 和存储方面的不足, 提出分支边界 skyline 查询算法 (branch-and-bound skyline, BBS). 还有研究数据流上的 skyline [13]、不确定的 skyline [14] 和基于组的 skyline [15] 等查询方法, 但是他们很少考虑到 skyline 查询中的隐私问题.

为解决查询中的数据机密性问题, 一些隐私保护技术被提出, 但并不适用于移动边缘计算场景下的面向位置信息的安全 skyline 查询. 其中, 私有信息检索 [16] 的查询是私有的, 但数据是公开的; 非对称点积保持加密 [17] 被证明在已知明文攻击下是不安全的; 保序加密 [18] 和 ORE 加密方案 [3] 无法进行有效的密文计算; 全同态加密 [19] 虽然可以获得较强的安全性, 但是它的计算开销较高. 针对安全 skyline 计算问题, 根据数据是否分布在多个云服务器上, 可将其分为分布式安全 skyline 查询和集中式安全 skyline 查询. 对于集中式安全 skyline 查询而言, 用户希望从存储外包数据的云服务器中获取 skyline 结果. Bothe 等 [20] 提出在加密数据库上支持 skyline 查询的方法, 但是没有提供正式的安全保证. 由于云服务器并不可信, Chen 等 [21] 专注于对 skyline 查询结果的验证. Liu 等 [10,11] 研究 skyline 计算的安全问题, 并提出基于加密数据的安全 skyline 查询协议 (BSSP), 由于其计算支配关系需要大量计算和通信开销导致效率不高. Wang 等 [22] 基于 ORE 加密方案提出动态 skyline 查询框架, 其效率较之 BSSP 提升了约 3 个数量级, 但是其无法进行距离计算.

对于分布式安全 skyline 查询而言, Liu 等 [23] 提出具有隐私保护性质的 skyline 计算方法, 由于位于本地服务提供商的数据未加密, 因此无法完全保证数据的机密性. 然后, Liu 等 [24] 提出在多个加密数据库上进行安全 skyline 查询的方法, 由于需要花费大量时间进行密文计算, 其效率并不高. Zaman 等 [25] 使用 MapReduce 实现分布式的安全 skyline 查询计算, 但是通信时间成为其 skyline 计算效率提升的瓶颈. 本文研究的是基于移动边缘计算场景下的安全 skyline 查询, 虽然其数据是分布于多台边缘服务器, 但是边缘侧更加注重查询的轻量化和效率. 因此, 以上安全 skyline 查询方法都在数据隐私或效率方面有所欠缺.

8 结论

针对传统云计算模型下安全 skyline 查询方案难以满足用户需求的问题, 本文主要研究移动边缘计算场景下基于位置信息的安全 skyline 查询协议. 在保证数据机密性和查询不可链接性的前提下, 本文设计了新颖的轻量级安全索引 SR*-tree, 以达到减少同态密文计算、优化索引存储的目的. 同时, 基于 SR*-tree 索引, 本文提出了平衡安全和效率的安全支配协议 STDM, 并进一步构建了面向位置信息的移动边缘计算安全 skyline 协议, 实现了云边协同的安全 skyline 计算. 实验在单边缘节点上对所提协议与现有的 BSSP 协议的查询效率进行了比较, 多个数据集上的实验结果表明本协议在具有相同安全性的条件下有着更高的查询效率. 此外, 实验通过对安全 skyline 查询进行并行化实现, 进一步评估了云边协同下安全 skyline 计算的效率.

参考文献

1 Liu Y H, Yang Q F, Li Z H. Cloud computing development environment: from code logic to dataflow diagram. Sci Sin Inform, 2019, 49: 1119–1137 [刘云浩, 杨启凡, 李振华. 云计算应用服务开发环境: 从代码逻辑到数据流图. 中国科学: 信息科学, 2019, 49: 1119–1137]

2 Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, 1999. 223–238

3 Lewi K, Wu D J. Order-revealing encryption: new constructions, applications, and lower bounds. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, 2016. 1167–1178

4 Yao A C C. How to generate and exchange secrets. In: Proceedings of Annual Symposium on Foundations of Computer Science, 1986. 162–167

5 Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: an online algorithm for skyline queries. In: Proceedings of the 28th International Conference on Very Large Data Bases, 2002. 275–286

6 Elmehdwi Y, Samanthula B K, Jiang W. Secure k-nearest neighbor query over encrypted data in outsourced environments. In: Proceedings of the 30th International Conference on Data Engineering, 2014. 664–675

7 Liu A, Zhengy K, Liz L, et al. Efficient secure similarity computation on encrypted trajectory data. In: Proceedings of the 31st International Conference on Data Engineering, 2015. 66–77

8 Liu X, Deng R H, Ding W, et al. Privacy-preserving outsourced calculation on floating point numbers. IEEE Trans Inform Forensic Secur, 2016, 11: 2513–2527

9 Börzsönyi S, Kossmann D, Stocker K. The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, 2001. 421–430

10 Liu J F, Yang J C, Xiong L, et al. Secure skyline queries on cloud platform. In: Proceedings of the 33rd International Conference on Data Engineering (ICDE), 2017. 633–644

11 Liu J F, Yang J C, Xiong L, et al. Secure and efficient skyline queries on encrypted data. IEEE Trans Knowl Data Eng, 2019, 31: 1397–1411

12 Papadias D, Tao Y F, Fu G, et al. Progressive skyline computation in database systems. ACM Trans Database Syst, 2005, 30: 41–82

13 Tao Y F, Papadias D. Maintaining sliding window skylines on data streams. IEEE Trans Knowl Data Eng, 2006, 18: 377–391

14 Liu J F, Zhang H Y, Xiong L, et al. Finding probabilistic k-skyline sets on uncertain data. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, 2015. 1511–1520

15 Yu W H, Qin Z, Liu J F, et al. Fast algorithms for Pareto optimal group-based skyline. In: Proceedings of ACM on Conference on Information and Knowledge Management, 2017. 417–426

16 Papadopoulos S, Bakiras S, Papadias D. Nearest neighbor search with strong location privacy. In: Proceedings of the VLDB Endowment, 2020. 619–629

17 Zhu Y W, Xu R, Takagi T. Secure k-NN computation on encrypted cloud data without sharing key with query users. In: Proceedings of International Workshop on Security in Cloud Computing, 2013. 55–66

18 Choi S, Ghinita G, Lim H S, et al. Secure kNN query processing in untrusted cloud environments. IEEE Trans Knowl Data Eng, 2014, 26: 2818–2831
19 Jiang B B, Zhang Y. Securely min and k-th min computations with fully homomorphic encryption. Sci China Inf Sci, 2018, 61: 058103
20 Bothe S, Karras P, Vlachou A. eSkyline: processing skyline queries over encrypted data. In: Proceedings of the 39th International Conference on Very Large Data Bases, 2013. 1338–1341
21 Chen W X, Liu M J, Zhang R, et al. Secure outsourced skyline query processing via untrusted cloud service providers. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications, 2016
22 Wang W G, Li H, Peng Y G, et al. Scale: an efficient framework for secure dynamic skyline query processing in the cloud. In: Proceedings of DASFAA, 2020. 288–305
23 Liu X M, Lu R X, Ma J F, et al. Efficient and privacy-preserving skyline computation framework across domains. Future Generation Comput Syst, 2016, 62: 161–174
24 Liu X M, Choo K-K R, Deng R H, et al. PUSC: privacy-preserving user-centric skyline computation over multiple encrypted domains. In: Proceedings of the 12th IEEE International Conference On Big Data Science and Engineering, 2018. 958–963
25 Zaman A, Siddique M A, Morimoto Y, et al. Secure computation of skyline query in MapReduce. In: Proceedings of International Conference on Advanced Data Mining and Applications, 2016. 345–360

Secure skyline query processing in mobile edge computing over location-based data

Zuan WANG[1], Xiaofeng DING[1*], Pan ZHOU[2*], Youliang TIAN[3] & Hai JIN[1]

1. *National Engineering Research Center for Big Data Technology and System, Service Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074*, China*;
2. *School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan* 430074*, China*;
3. *College of Computer Science and Technology, Guizhou University, Guiyang* 550025*, China*
* Corresponding author. E-mail: xfding@hust.edu.cn, panzhou@hust.edu.cn

Abstract Aiming at the efficiency and privacy of queries under mobile edge computing, secure skyline query processing in mobile edge computing over location-based data is developed. A secure skyline query framework in the mobile edge computing scenario is first proposed and a security model is also defined. To improve the efficiency of secure queries in the edge server, we use the Paillier homomorphic encryption and order-revealing encryption to devise a novel and unified secure index structure. It can effectively protect the query unlinkability in accordance with the semi-blind structure proposed by us. Furthermore, based on the secure index, we propose a secure skyline query protocol in mobile edge computing to realize the edge-cloud collaborative skyline query considering the privacy leakage. Finally, we analyze the complexity and security of the proposed approach. Findings from the experimental evaluation show that our proposed approach significantly increases the efficiency under the semi-honest model, in comparison to the state-of-art.

Keywords secure skyline queries, location-based data, mobile edge computing, secure index, semi-honest model

Zuan WANG was born in 1992. He received his M.S. degree in computer science and technology from Guizhou University, Guiyang, China, in 2019. Currently, he is a Ph.D. student in the School of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan, China. His current research interests include information security, data privacy, and query processing.

Youliang TIAN was born in 1982. He received his B.S. and M.S. degrees in mathematics from Guizhou University, Guiyang, China, in 2004 and 2009, respectively, and his Ph.D. degree in cryptography from Xidian University, Xian, China, in 2012. Currently, he is a professor and Ph.D. supervisor at the College of Computer Science and Technology, Guizhou University. His current research interests include algorithmic game theory, cryptography and security protocols, big data security and privacy protection, blockchain, electronic currency.

Xiaofeng DING was born in 1982. He received his Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China in 2009. Currently, he is working as an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests mainly include data privacy and query processing, data encryption, graph databases and crowdsourcing.

Hai JIN was born in 1966. He received his Ph.D. degree in computer engineering from the Huazhong University of Science and Technology (HUST), China, in 1994. He has worked with the University of Hong Kong, from 1998 to 2000, and a visiting scholar with the University of Southern California, from 1999 to 2000. He is currently the Cheung Kung scholars chair professor of computer science and engineering with HUST. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security.