# 结构化加密图的最短路径查询

潘瑛颖[1,2], 陈兰香[1,2]

[1]福建师范大学 计算机与网络空间安全学院 福州 中国 350117
[2]福建省网络安全与密码技术重点实验室 福州 中国 350117

**摘要** 随着云计算的快速发展, 数据用户将大量图数据外包给云以节约存储和管理成本。然而, 外包数据的安全隐私问题是云计算面临的一大挑战。由于云是半诚实的, 为保护敏感信息的隐私安全, 数据拥有者希望在将图数据外包给云服务器之前对其加密, 同时保留对加密的图数据进行查询和处理的能力。最短路径查询查找图中给定两节点之间的最短路径, 是图应用中最基础的查询类型之一。目前已有许多研究者提出一系列高效的方案, 以支持加密图上近似或精确最短距离查询、约束最短距离查询和 top-*k* 最近关键字查询, 但支持最短路径查询的方案较少, 且已有方案的存储与时间开销较大。本文提出一种支持在加密图上进行两节点间最短路径查询的结构化加密图方案。在本方案中, 我们基于 2-Hop 标签技术构造支持有向图上最短路径查询的标签索引并加密, 然后将加密的标签外包给云服务器。利用改进的保序编码算法编码距离值, 实现加法运算和值的比较, 提高最短路径查询的效率。在查询阶段, 通过递归式地计算两节点间最短路径上的第一条边和最后一条边, 最终输出完整的最短路径。安全性和性能分析证明本文方案是安全有效的, 能以较小的存储和较高的查询效率实现两节点间的最短路径查询并保护图数据的隐私。

**关键词** 云计算; 图加密; 结构化加密; 最短路径查询
**中图法分类号** TP309.7 DOI 号 10.19363/J.cnki.cn10-1380/tn.2024.07.05

# Shortest Path Queries on Structured Encrypted Graph

PAN Yingying[1,2], CHEN Lanxiang[1,2]

[1] College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China
[2] Fujian Provincial Key Laboratory of Network Security and Cryptology, Fuzhou 350117, China

**Abstract** With the rapid development of cloud computing, data users outsource large amounts of graph data to the cloud to save the cost of data storage and management. However, the security and privacy of outsourced data is a major challenge for cloud computing. As the cloud is semi-honest, in order to protect the privacy of sensitive information, data owners want to encrypt the graph data before outsourcing it to cloud servers, while retaining the ability to query and process the encrypted graph data. Shortest path query, one of the most fundamental query types in graph application, retrieves the shortest path between two given nodes in the graph. At present, many researchers have proposed a series of efficient schemes supporting approximate or accurate shortest distance query, constrained shortest distance query and top-*k* nearest keyword query on encrypted graph. However, there are fewer schemes supporting shortest path query, and the storage and time overhead of existing schemes are large. In this paper, a structured encrypted graph scheme is proposed to support shortest path query between two given nodes on the encrypted graph. In this scheme, we generate and encrypt a label index that supports the shortest path query on the directed graph based on 2-Hop labeling, and then outsource the encrypted index to the cloud server. A modified order preserving encoding algorithm is utilized to encode the distance value to achieve the addition operation and comparison of values, improving the efficiency of the shortest path query. The first edge and the last edge on the shortest path between two given nodes are calculated recursively during the query process, and finally the complete shortest path is returned. Security and performance analysis prove that the proposed scheme is secure and efficient to achieve the shortest path query between two nodes with less storage overhead and higher query efficiency, and protect the privacy of graph data.

**Key words** cloud computing; graph encryption; structured encryption; shortest path query

## 1 引言

长期以来, 图数据被应用于科研和工程的许多领域, 如社交网络、知识图谱、生物识别网络、道路网、通信网络等。为有效地分析和处理图数据, GraphLab[1]、Pregel[2]、TurboGraph[3]等一系列工具被

**通讯作者:** 陈兰香, 工学博士, 教授, 博士生导师, Email: lxiangchen@fjnu.edu.cn。
本课题得到国家自然科学基金(No. 62072105)和国家自然科学基金海峡联合基金重点项目(No. U1805263)资助。
收稿日期: 2022-11-05; 修改日期: 2023-02-13; 定稿日期: 2024-03-15

提出用以查询、管理和分析大规模图数据。

随着云计算深入人们的生活与工作, 个人和企业越来越倾向于将数据外包给云进行存储和计算。图数据拥有者通过将大规模图外包给第三方的云服务器, 节约数据的管理和维护成本, 但与此同时增加了数据隐私泄露的风险。因此, 通常需要在将图数据外包给云服务器之前对其加密, 同时保留一定的数据查询和处理能力。

2010 年, Chase 和 Kamara[4]引入结构化加密(Structured Encryption, STE)的概念, 将应用于文本数据的对称可搜索加密(Symmetric Searchable Encryption, SSE)推广到任意结构的数据, 使数据能够通过特殊的令牌进行秘密查询。图加密是结构化加密的一个典型应用, Chase 和 Kamara[4]首先设计和提出支持加密图的邻居查询、邻接查询和标签图上聚集子图查询的结构化加密方案。使用这些方案, 图数据的拥有者能将加密的图数据外包给半诚实的云服务器而保持查询数据的能力。此后, 一系列支持近似或精确的最短距离查询[5-6], 约束最短距离查询[7-8], top-$k$ 最近关键字(top-$k$ Nearest Keywords, $k$NK)查询[9]的图加密方案被先后提出。

最短路径查询是图应用中最基本的查询类型之一, 在图计算中用途广泛。两节点间的最短路径查询以图的两个节点作为输入, 搜索两节点间的最短路径。在道路网中, 最短路径查询返回两个地点间的距离最短的路线; 在社交网络中, 最短路径查询返回两个用户间的最短联系链。单源最短路径(Single Source Shortest Path, SSSP)算法如 Dijkstra 算法[10]和 Bellman-Ford 算法[11]是处理该问题的经典方法。但在加密图上执行 SSSP 算法实时计算两节点间最短路径在实际应用中存在一些困难。为提高在大规模图上进行最短路径查询的效率, 可以对图进行预处理, 构建合适的数据结构用于高效回答两节点间最短路径查询, 2-Hop 标签[12]是解决此类问题的方法之一。然而, 基本的 2-Hop 距离标签不能很好地直接应用于最短路径查询中, 需要进行较为复杂的迭代过程, 不断地找出路径上的节点并为其进行排序; 在已有的图距离或路径查询方案中, 其计算方法通常基于同态加密, 导致方案的计算和通信开销都比较大。

本文针对有向图提出支持两节点间最短路径查询的结构化加密图方案。两节点间的一条最短路径可以视作一组有序边的集合, 本文方案在 2-Hop 距离标签的基础上构造了一个支持最短路径查询的 2-Hop 标签索引, 并且由于该索引不需要使用实际的最短距离值, 采用改进的保序编码(Order Preserving Encoding, OPE)[9]对距离值进行加密, 较好地提高了计算与通信效率。通过递归依次求解最短路径上的第一条和最后一条边, 最终返回两节点间的最短路径。

(1)具体而言, 本文提出一种支持两节点间最短路径查询的结构化加密图方案, 实现以下目标:

提出一个高效安全的结构化加密图方案, 支持有向图上的最短路径查询。本方案可以在加密图数据上实现隐私保护的最短路径查询, 回答图中任意两个节点间的最短路径问题。

(2)本方案基于 2-Hop Cover 概念构造图的 2-Hop 标签索引实现加密有向图上最短路径查询, 提高大规模图上最短路径查询的效率。通过在 2-Hop 标签索引的生成过程中将无向图的每一条边视作两条方向相反的有向边, 该方案可以自然地扩展到无向图。

(3)本方案在真实数据集上进行测试, 与已有的加密方案相比, 本方案的存储空间开销较小而最短路径查询效率也较高, 能在存储空间和查询效率之间取得较好的平衡。

下文章节组织如下: 第 2 节介绍最短路径查询及加密图数据查询的相关工作; 第 3 节定义符号并介绍方案的系统模型与安全模型; 第 4 节介绍本方案使用的主要工具及预备知识; 第 5 节详细介绍本文提出的结构化加密图的最短路径查询方案的算法; 第 6 节和第 7 节对方案进行安全性分析和性能评估; 第 8 节讨论有待进一步研究的方向; 最后, 总结全文并指出下一步的工作方向。

## 2 相关工作

本节从明文图数据的最短路径查询与加密图上的数据查询两个方面进行介绍。

### 2.1 明文图数据最短路径查询

SSSP 算法如 Dijkstra 算法[10]和 Bellman-Ford 算法[11]是处理两节点间最短路径查询的经典方法。但是, Dijkstra 算法和 Bellman-Ford 算法的时间复杂度分别为 $O(m+n\log n)$和 $O(nm)$[11], 在大规模图上收到查询请求后执行 SSSP 算法计算两节点间最短路径通常需要超过 1 s, 对于需要实时交互的应用来说可能无法满足实际应用的需求[13]。因此许多研究者考虑对图进行预处理, 构建合适的索引获得图的紧凑表示, 如传递闭包[11]和 2-Hop 标签[12], 用于加速查找两节点间的最短路径。

Cohen 等人[12]提出一种名为 2-Hop 标签的索引结构, 通过在两节点对应的标签集中找到具有共享节点的项, 回答可达性或距离查询。可以向标签的每

一项添加相应路径上的第一条边(first-edge)信息, 通过递归查询第一条边求取最短路径。Akiba 等人[13]采用基于剪枝的宽度优先搜索(Breadth First Search, BFS)的剪枝地标标签法(Pruned Landmark Labeling, PLL)构造 2-Hop 标签索引, 在大规模网络上快速精确地计算两节点间的最短距离, 有效减小索引规模, 提高索引构建效率。Wang 等人[14]提出框架查询(Query-by-Sketch)方案, 利用预先计算的离线标签回答最短路径图查询, 返回两节点间所有最短路径。Liu 等人[15]提出一种基于 skyline 路径连接的 CSP-2-Hop 标签方法并进一步提出森林 Hop 标签, 在更大的网络上实现无向图上精确约束最短路径查询。Farhan 等人[16]引入跳跃和修剪搜索策略同时用于标签的增量算法和减量算法, 在边插入或边删除后高效更新 Highway Cover 标签, 实现大规模图距离查询的全动态标签方法的构造。

然而, 上述方法都是基于明文图数据实现, 用户数据往往包含大量敏感信息。为保护隐私, 用户希望在将图数据存储到云之前对其进行加密, 但是数据加密将使查询与处理变得困难, 如何对加密图数据进行查询和处理是一大挑战。

## 2.2 加密图数据查询

2000 年, Song 等人[17]首先提出对称可搜索加密, 允许用户对存储在半诚实服务器上的加密数据进行检索。此后, 研究者提出一系列密文检索方法[17-19]。由于普通用户本地资源有限, 随着云计算的普及, 大型图数据被外包给云成为趋势, 图的隐私保护日益受到重视。

2010 年, Chase 和 Kamara[4]首先提出支持邻居查询、邻接查询和关键词聚集子图查询的结构化加密方案。Cao 等人[20]设计基于特征的索引和用于过滤的安全内积计算, 解决加密图结构的隐私保护查询问题。Yin 等人[21]提出隐私保护的 2-Hop 标签方案, 实现加密图数据的可达性查询。2015 年, Meng 等人[5]基于对距离预言机(Distance Oracle)的加密设计了三种具有不同计算和通信效率的图加密方案, 用于回答无向图上近似最短距离查询。2021 年, Liu 等人[22]提出一种基于 2-Hop Cover 标签的图加密方案 GENOA, 以泄露部分距离值顺序信息为代价完成最短距离查询。Liu 等人[22]构造了一个包括顺序编码、间隔编码和对称密钥加密三部分的保序加密方案对标签中的距离值加密。受限于距离值的加密方法, GENOA[22]返回的最短距离可能存在误差, 但与 Meng 等人[5]的方案相比, 由于 GENOA 不使用公钥加密, 且借助保序加密方案对候选结果集进行过滤和选择, 具有更高的效率和准确度。

针对有向图的约束最短距离(Constrained Shortest Distance, CSD)查询, Shen 等人[7]提出一个回答加密图上的近似 CSD 查询的方案 Connor, 并基于揭序加密(Order Revealing Encryption, ORE)设计密文比较协议筛选成本值。此后, Zhang 等人[8]设计并实现了支持有向图上精确 CSD 查询的图加密方案, 方案利用门限解密的 Paillier 加密系统(Paillier Cryptosystem with Threshold Decryption, PCTD)加密距离和成本值, 并基于 PCTD 设计安全比较协议在加密条件下完成整数的比较, 最终返回满足约束条件的最小距离值。但该比较协议需要云存储服务器与不合谋的多个云计算服务器协同完成。此外, Liu 等人[9]提出一个支持 top-$k$ 最近关键字查询(top-$k$ Nearest Keywords, $k$NK)的图加密方案, 查询距离给定节点最近的 $k$ 个带有给定关键词的节点。方案构造了关键词索引、基于 2-Hop 距离标签的 Hop 索引、存储节点邻居的邻居索引和检查节点是否包含某个关键词的查找索引对标签图数据进行加密, 混合使用基于关键词索引和 Hop 索引的前向搜索与基于邻居索引和查找索引的后向搜索两种搜索策略提高查询效率。Liu 等人[9]还提出一种改进的保序编码算法(Order-Preserving Encoding, OPE)对 2-Hop 标签中的距离值进行加密, 可以完成一次加法操作后值的比较。Sun 等人[23]进一步提出加密图上约束 top-$k$ 最近模糊关键字查询方案。Guan 等人[24]在双服务器环境下针对二部图使用对称同态加密方法加密边表和节点表索引与查询请求, 提出两种具有不同安全级别的($\alpha$, $\beta$)-核心查询方案。

2013 年, Aly 等人[25]首次使用安全多方计算技术解决传统的最短路径问题的安全计算。2015 年, Samanthula 等人[26]分别在单云和联邦云环境下提出两种具有不同安全性和效率的协议实现加密图的最短路径查询。Wang 等人[6]提出可动态更新的图数据库加密方案 SecGDB, 对图本身的邻接表进行加密, 结合加法同态加密和混淆电路借助代理服务器在加密图上执行 Dijkstra 算法, 计算两节点间的最短距离和路径。此外, SecGDB 构造一个更新字典用于边的增加或删除, 并设计了一个辅助数据结构用于存储查询历史, 快速返回重复查询的结果。Ramezanian 等人[27]扩展 Floyd-Warshall 算法生成扩展的传递闭包矩阵, 存储最短路径上终点的前驱节点信息并使用 AES 加密, 他们还构造了一个密钥矩阵存储传递闭包各个位置的 AES 密钥并运用 RSA 加密, 使用 Paillier 加密方案构建隐私信息检索(Private Information Retrieval, PIR)协议隐藏查询, 提出一个有向图

上的最短路径查询方案。Bramm 等人[28]设计了基于 Paillier 和 ElGamal 的交互式同态保序加密方案 ehOPE, 结合 ehOPE 和结构化加密提出一个通用框架加密图数据, 并应用该框架实现加密图上的 Dijkstra 算法。以上三种方案都针对有向图提出, 但可以扩展到无向图。同时, 上述方案均使用同态加密完成计算和比较, 计算效率有待提升。2021 年, Ghosh 等人[29]根据结构化加密的基本框架提出高效的最短路径查询的图加密方案 GES。GES 预计算任意两节点间最短路径并生成最短路径矩阵 SP-martix 记录最短路径上第一条边, 根据 SP-martix 构造加密字典。给定起点和终点, 通过递归获取路径上的下一条边直到输出完整路径。该方案适用于任意可在明文下计算出最短路径的图, 查询时间开销只与路径边数有关。

与以上方案不同, 本文提出一个结构化加密图上的最短路径查询方案, 在加密条件下查询有向图上两节点间的最短路径。在技术上, 本文基于 2-Hop 距离标签构造一个支持最短路径查询的图标签索引并加密, 与文献[26-28]的方案不同, 本文不使用同态加密, 而是使用 Liu 等人[9]提出的改进的保序编码算法 OPE 对距离值进行编码, 进行加法运算与值的比较。在功能上, 不同于文献[9]提出的 *k*NK 查询方案和文献[22]提出的最短距离查询方案, 本文在加密图标签上通过递归式地计算两节点最短路径上的第一条和最后一条边, 最终返回两节点间的最短路径。对于无向图, 本文方案可以将每一条无向边视作两条方向相反的有向边, 运行 PLL 算法生成 2-Hop 标签, 从而将方案自然地扩展到无向图。

## 3 符号定义与系统模型

### 3.1 符号定义

本文使用的符号定义如表 1 所示。两个伪随机函数 $h$ 和 $g$ 表示如下:

- 伪随机函数 $h:\{0,1\}^{\lambda}\times\{0,1\}^{*}\to\{0,1\}^{\lambda}$;
- 伪随机函数 $g:\{0,1\}^{\lambda}\times\{0,1\}^{*}\to\{0,1\}^{3\lambda+l_1+l_2}$;

其中, $\lambda$ 是安全参数, $l_1$ 和 $l_2$ 分别是距离值的保序编码长度和对称加密方案 SKE 的密文长度。

对称密钥加密方案 SKE = (Gen, Enc, Dec)由 3 个多项式时间算法组成。密钥生成算法 SKE.Gen 以安全参数 $\lambda$ 为输入, 返回对称密钥 $K$; 加密算法 SKE.Enc 以密钥 $K$ 和明文消息 $m$ 为输入, 返回密文 $c$; 解密算法 SKE.Dec 以密钥 $K$ 和密文 $c$ 为输入, 如果 $K$ 是密文 $c$ 的正确密钥, 则返回 $m$。SKE 在选择明文攻击下具有伪随机的密文, 满足 CPA 安全。

**表 1 符号定义**

**Table 1 Symbol definition**

| 符号 | 说明 |
|---|---|
| OPE | 保序编码算法 |
| SKE | 对称密钥加密方案 |
| $G=(V, E)$ | 有向图, 包括顶点集 $V$ 和边集 $E$ |
| $\|V\|, \|E\|$ | 顶点集 $V$ 和边集 $E$ 的基数 |
| $\lambda$ | 安全参数 |
| $K$ | 密钥集 |
| $K_1$ | 计算加密标签索引入口的密钥 |
| $K_2$ | 保序编码算法 OPE 的密钥 |
| $K_3$ | 对称密钥加密方案 SKE 的密钥 |
| $L=\{L_{\text{out}}, L_{\text{in}}\}$ | 2-Hop 标签集, 包括 out 标签和 in 标签 |
| $EL=\{EL_{\text{out}}, EL_{\text{in}}\}$ | 加密的图标签集, 包括 out 标签和 in 标签 |
| $L(v)=\{L_{\text{out}}(v), L_{\text{in}}(v)\}$ | 与节点 $v$ 关联的 2-Hop 标签 |
| $EL(v)=\{EL_{\text{out}}(v), EL_{\text{in}}(v)\}$ | 与节点 $v$ 关联的加密标签 |
| $I_v^{+}, K_v^{+}$ | 与 $v$ 关联的 out 标签项的索引入口 |
| $I_v^{-}, K_v^{-}$ | 与 $v$ 关联的 in 标签项的索引入口 |
| $d_{vu}$ | 从节点 $v$ 到 $u$ 的最短距离 |
| $D_{vu}$ | 从节点 $v$ 到 $u$ 的最短距离的保序编码值 |
| $<v, u>$ | 以 $v$ 为始点、$u$ 为终点的有向边 |
| $q=(s, t)$ | 查询请求, 由起点 $s$ 和终点 $t$ 组成 |
| $\tau_{st}$ | 查询从起点 $s$ 到终点 $t$ 的最短路径的令牌 |
| $\perp$ | 空 |
| $R$ | 加密的查询结果 |
| $p$ | 解密的最短路径明文结果 |

### 3.2 系统模型

本文提出的支持最短路径查询的结构化加密图方案系统模型包括 3 个实体: 数据拥有者(Data Owner)、云服务器(Cloud Server)和数据使用者(Data User), 如图 1 所示。

数据拥有者是图数据的拥有者, 持有原始图 $G$。数据拥有者生成所需的密钥集 $K=\{K_1, K_2, K_3\}$, 生成支持最短路径查询的 2-Hop 标签并使用密钥集 $K$ 加密, 获得加密的图标签 $EL$ 并发送给云服务器。当收到数据使用者的查询请求, 数据拥有者生成查询令牌 $\tau_{st}$, 将令牌 $\tau_{st}$ 和用于解密密文结果 $R$ 的密钥传输给数据使用者。

云服务器是存储和计算服务的提供者。云服务器存储加密的图标签数据, 根据数据使用者的查询请求在加密数据上进行查询并向数据使用者返回密文结果 $R$。

数据使用者是一个想要在加密的图数据上执行查询的用户。数据使用者向数据拥有者发送查询请求 $q=(s, t)$获得查询令牌, 然后发送令牌给云服务器请求查询, 最后对获得的密文结果 $R$ 进行解密。

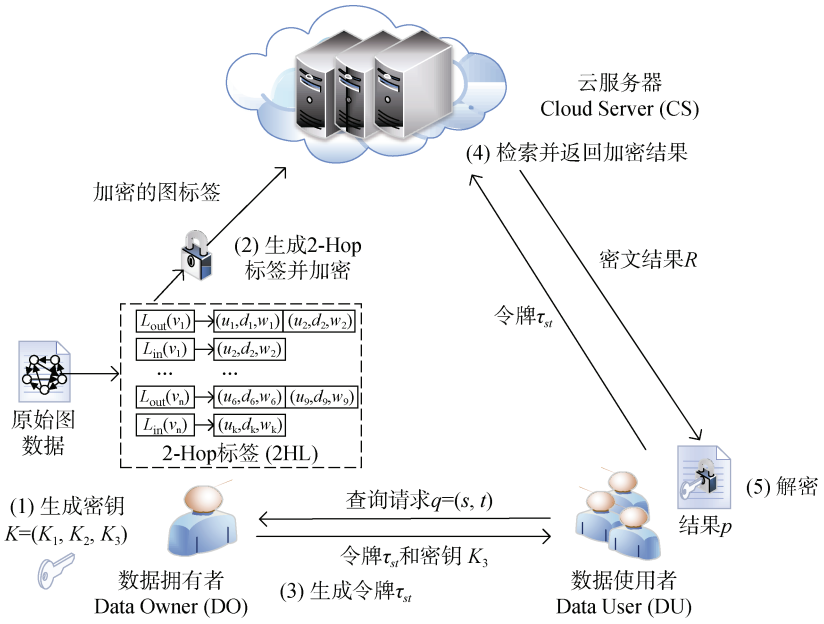



**图 1　系统模型**

**Figure 1　System model**

本文提出一个支持有向图两节点节最短路径查询的结构化加密图方案, 对于该结构化加密图方案定义如下:

**定义 1.(结构化加密图方案)**支持两节点间最短路径查询的结构化加密图方案是一个包括 5 个算法的五元组 Π = (KeyGen, GraphEnc, Token, PathQuery, Decrypt), 每个算法形式化定义如下:

(1)$K\leftarrow$KeyGen($1^{\lambda}$): 密钥生成算法。一个由数据拥有者运行的概率算法, 输入安全参数 $\lambda$, 输出密钥集 $K=\{K_1, K_2, K_3\}$;

(2)$EL\leftarrow$GraphEnc($K, G$): 图的加密算法。概率算法, 数据拥有者以密钥集 $K$ 和图 $G=(V, E)$作为输入, 输出加密的图标签索引 $EL$;

(3)$\tau_{st}\leftarrow$Token($K, q$): 令牌生成算法。由数据拥有者运行的确定性算法, 以密钥集 $K$ 和查询 $q=(s, t)$为输入, 输出令牌 $\tau_{st}$, $s$ 和 $t$ 分别是待查询最短路径的起点和终点;

(4)$R\leftarrow$PathQuery($EL, \tau_{st}$): 最短路径查询算法。一个由云服务器运行的确定性算法, 以加密的图标签 $EL$ 和令牌 $\tau_{st}$ 为输入, 根据 $\tau_{st}$ 在 $EL$ 上查找和计算节点 $s$ 到 $t$ 的最短路径, 返回加密的结果 $R$;

(5)$p\leftarrow$Decrypt($K_3, R$): 解密算法。由数据使用者输入授权的对称密钥 $K_3$ 和云服务器返回的加密结果 $R$, 解密得到最短路径明文结果 $p$。

## 3.3　安全模型

### 3.3.1　泄露函数

首先, 我们对泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 进行介绍。

**泄露函数 $\mathcal{L}_1$**: 泄露函数 $\mathcal{L}_1$ 揭示可以从存储在云服务器上的加密标签 $EL$ 中推断出的信息, 包括图 $G$ 的节点数$|V|$, 图标签 $L_{\text{out}}$ 和 $L_{\text{in}}$ 的总项数$|L_{\text{out}}|$和$|L_{\text{in}}|$。距离值的顺序信息并不包含在内, 因为标签中的每一项数据都在加密后与一个伪随机函数值异或进行混淆。因此, 泄露函数 $\mathcal{L}_1=\{|V|, |L_{\text{out}}|, |L_{\text{in}}|\}$。

**泄露函数 $\mathcal{L}_2$**: 泄露函数 $\mathcal{L}_2$ 描述服务器接收令牌进行最短路径查询过程发生的各种泄露, 即云服务器从收到的令牌和运行 PathQuery 算法过程中收集和推断得到的信息, 包括查询模式泄露 $\mathcal{L}_{\text{QP}}(\boldsymbol{q})$, 标签模式泄露 $\mathcal{L}_{\text{LP}}(EL, \boldsymbol{q})$, 路径交叉模式泄露 $\mathcal{L}_{\text{PIP}}(EL, \boldsymbol{q})$ 和结果路径包含的边数 $l$。因此, 泄露函数 $\mathcal{L}_2=\{\mathcal{L}_{\text{QP}}(\boldsymbol{q}), \mathcal{L}_{\text{LP}}(EL, \boldsymbol{q}), \mathcal{L}_{\text{PIP}}(EL, \boldsymbol{q}), l\}$。$\mathcal{L}_{\text{QP}}(\boldsymbol{q})$、$\mathcal{L}_{\text{LP}}(EL, \boldsymbol{q})$ 和 $\mathcal{L}_{\text{PIP}}(EL, \boldsymbol{q})$ 具体描述如下:

(1)查询模式泄露 $\mathcal{L}_{\text{QP}}(\boldsymbol{q})$: 查询模式揭示当前查询与历史查询是否存在重复。设 $\boldsymbol{q}=\{q_1, \ldots, q_m\}$是一个非空的查询集, 任意查询 $q_i\in\boldsymbol{q}$ 可解析为对应的元组$(s_i, t_i)$。对于任意两个查询 $q_i, q_j\in\boldsymbol{q}$, 定义 Sim$(q_i, q_j)$=$(s_i=s_j, t_i=t_j)$, 即是否$q_i=(s_i, t_i)$中的每个元素都和$q_j=(s_j, t_j)$匹配。查询模式泄露函数 $\mathcal{L}_{\text{QP}}(\boldsymbol{q})$ 将返回一个 $m\times m$ 的矩阵, 第 $i$ 行第 $j$ 列的项为 Sim$(q_i, q_j)$。由于对节点真实标识符使用伪随机函数进行隐藏, $\mathcal{L}_{\text{QP}}(\boldsymbol{q})$ 不泄露查询顶点的真实标识符。

(2)标签模式泄露 $\mathcal{L}_{\text{LP}}(EL, \boldsymbol{q})$: 标签模式泄露与待查询节点 $v$ 关联的标签 $L_{\text{out}}(v)$或 $L_{\text{in}}(v)$的部分信息,

如两个不同标签集之间的部分共享节点、距离值大小顺序和标签大小信息。已知 $EL$ 是图 $G$ 的加密的 2-Hop 标签, 对于给定查询 $q=(s, t)$, 定义标签模式泄露函数 $\mathcal{L}_{\mathrm{LP}}(EL,\boldsymbol{q})=(\sum,Z)$。$\sum$ 是与查询节点关联的加密标签。具体而言, 对于出现在历史查询中的节点集合 $\{v_1^*,\cdots,v_k^*\}$, $\sum=\{EL(v_1^*),\cdots,EL(v_k^*)\}$。$Z=(X, Y)$, 其中 $X=\{h(v):(v, d, w)\in L_{\mathrm{out}}(s)\}$和 $Y=\{h(v):(v, d, w)\in L_{\mathrm{in}}(t)\}$是多重集, $v$ 泄露的是其伪随机标识符, 敌手无法获得相应节点的真实标识符。由于对距离值使用保序编码, 标签模式泄露距离值的顺序信息。

(3)路径交叉模式泄露 $\mathcal{L}_{\mathrm{PIP}}(EL,\boldsymbol{q})$: 路径交叉模式泄露 $\mathcal{L}_{\mathrm{PIP}}(EL,\boldsymbol{q})$ 泄露两条路径的交叉的节点及其在路径上排序的相关信息, 本文主要在数据项索引入口信息部分中泄露。假设 $p_{st}$ 是节点 $s$ 出发到节点 $t$ 的最短路径, 对于所有历史查询 $q_i=(s_i, t_i)$, 对应最短路径为 $p_{s_it_i}$, 如果存在 $s_i=s$ 或 $t_i=t$, 那么 $\mathcal{L}_{\mathrm{PIP}}(EL,\boldsymbol{q})$ 泄露 $p_{s_it_i}\cap_r p_{st}$, 详细定义见文献[29]。

### 3.3.2 本文方案的安全模型

本方案中假设云服务器是诚实但好奇的, 即云服务器会根据协议对查询请求执行相应的操作并返回查询结果, 但同时会试图在执行过程中获取关于图数据的隐私信息。本文提出一个支持最短路径查询的结构化加密图方案, 以加密图数据并实现最短路径查询, 我们提出的方案是一种结构化加密方案。本方案基于文献[4, 30]中提出的自适应选择查询攻击(Adaptive Chosen Query Attacks)安全性定义, 即 CQA2-安全性定义。在本文的模型中, 云服务器被认为是敌手 $\mathcal{A}$。根据云服务器在数据存储和接收令牌进行查询过程中掌握的信息给出泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$。

**定义 2.(CQA2-安全性)** $\Pi$ = (KeyGen, GraphEnc, Token, PathQuery, Decrypt)是一个结构化加密图方案, $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 是方案的两个泄露函数。给定安全参数 $\lambda$, (半诚实)敌手 $\mathcal{A}$ 和模拟器 $\mathcal{S}$ 执行的两个概率性实验 $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ 和 $\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ 定义如下:

(1) $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$: 敌手 $\mathcal{A}$ 选择一个待加密的图 $G$ 发送给挑战者。挑战者运行 KeyGen($1^\lambda$)生成密钥 $K$, 运行 GraphEnc($K$, $G$)算法计算加密标签 $EL$ 返回给 $\mathcal{A}$。$\mathcal{A}$ 生成多项式数量的自适应查询 $\boldsymbol{q}=\{(s_1, t_1), \ldots, (s_n, t_n)\}$发送给挑战者。对于每个查询 $q_i=(s_i, t_i)\in\boldsymbol{q}$, 挑战者运行 Token($K$, $q_i$)生成对应的令牌 $\tau_i$ 发送给 $\mathcal{A}$, 敌手 $\mathcal{A}$ 运行 PathQuery($EL$, $\tau_i$)获得 $R$。最后, 敌手 $\mathcal{A}$ 返回一个 bit 位 $b\in\{0,1\}$作为 $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ 实验的输出。

(2) $\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$: 敌手 $\mathcal{A}$ 选择一个待加密的图 $G$。给定泄露函数 $\mathcal{L}_1(G)$, 模拟器 $\mathcal{S}$ 模拟得到加密标签 $EL^*$并发送给敌手 $\mathcal{A}$。敌手 $\mathcal{A}$ 生成多项式数量的自适应查询 $\boldsymbol{q}=\{(s_1, t_1), \ldots, (s_n, t_n)\}$。给定泄露函数$\mathcal{L}_2$, 对于每个查询 $q_i=(s_i, t_i)\in\boldsymbol{q}$, 模拟器 $\mathcal{S}$ 获得$\mathcal{L}_2(G, q_i)$, 通过 $\mathcal{L}_2(G, q_i)$模拟令牌 $\tau_i^*$ 返回给 $\mathcal{A}$, $\mathcal{A}$ 运行 PathQuery($EL^*$, $\tau_i^*$)获得模拟查询结果 $R^*$。最后, 敌手 $\mathcal{A}$ 返回一个 bit 位 $b\in\{0,1\}$作为 $\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ 实验的输出。

如果对于所有概率多项式时间(Probabilistic Polynomial Time, PPT)敌手 $\mathcal{A}$, 存在一个 PPT 模拟器 $\mathcal{S}$, 满足:

$$|\Pr[\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)=1]-\Pr[\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)=1]|\leqslant \mathrm{negl}(\lambda)$$

其中, negl($\lambda$)是一个可忽略函数, 则方案 $\Pi$ 对于自适应选择查询攻击是 $(\mathcal{L}_1,\mathcal{L}_2)$-安全的。

## 4 预备知识

### 4.1 2-Hop 标签

根据图计算并生成用于查询的索引, 可以用于回答两节点间的最短路径。本文选择基于 2-Hop 标签方法(2-Hop Labeling, 2HL)构造支持两节点间最短路径查询的标签。2-Hop 标签[12-13]是一种预计算的可用于加速查找最短路径过程的索引结构。本文方案主要考虑有向图上的最短路径查询, 但是我们可以将无向图的每一条边视作一对方向相反的有向边来计算该图的 2-Hop 标签索引, 从而使方案的适用范围自然扩展到无向图。

#### 4.1.1 2-Hop 距离标签

有向图的 2-Hop 距离标签基本概念如下: 给定有向图 $G=(V, E)$, $L$ 是图 $G$ 的 2-Hop 标签。$L$ 为每个节点 $v$ 分配标签集 $L(v)=\{L_{\mathrm{out}}(v), L_{\mathrm{in}}(v)\}$。$L_{\mathrm{in}}(v)$是$(u, d_{uv})$对的集合, $d_{uv}$ 表示从 $u$ 出发到 $v$ 的最短距离; $L_{\mathrm{out}}(v)$是$(u, d_{vu})$对的集合, $d_{vu}$ 表示从 $v$ 到 $u$ 的最短距离。

2-Hop 距离标签确保对于任意两个可达的顶点 $s$ 和 $t$, 存在至少一个节点 $u$, 满足: (1)$(u, d_{su})\in L_{\mathrm{out}}(s)$, $(u, d_{ut})\in L_{\mathrm{in}}(t)$, (2)$u$ 在从 $s$ 到 $t$ 的最短路径上。

因此, 对于任意两个可达节点 $s$ 和 $t$, 可以使用以下公式在标签集 $L$ 上计算和回答从起点 $s$ 到终点 $t$ 之间的最短距离查询 distQuery($s$, $t$, $L$):

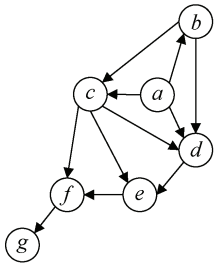
$$\mathrm{distQuery}(s, t, L)=\min\{d_{su}+d_{ut}\} \quad (1)$$

其中, $(u, d_{su})\in L_{\mathrm{out}}(s)$且$(u, d_{ut})\in L_{\mathrm{in}}(t)$。如果 $L_{\mathrm{out}}(s)$和 $L_{\mathrm{in}}(t)$没有共享的节点 $u$, 即 $s$ 到 $t$ 不可达则定义 $\mathrm{distQuery}(s,t,L)=\infty$。如果对有向图 $G$ 上任意一对节点 $s$ 和 $t$, 从 $s$ 到 $t$ 的最短距离 $d_{st}$=distQuery($s$, $t$, $L$)

成立, 则认为 $L$ 是 $G$ 的一个 2-Hop Cover。

Akiba 等人[13]提出的剪枝地标标签(Pruned Landmark Labeling, PLL)算法是一种有效的 2-Hop 标签计算方法, 通过按某种顺序依次对图中每个节点沿正向边和反向边方向执行剪枝的 BFS, 最终获得图 G 的 2-Hop 标签。假设 $L^{k-1}$ 前 $k$-1 个节点完成剪枝的 BFS 时得到的标签集, $d[u]$是从第 $k$ 个节点 $v_k$ 沿正向边出发访问节点 $u$ 时经过的路径长度, 如果 distQuery($v_k$, $u$, $L^{k-1}$)>$d[u]$, 将($v_k$, $d[u]$)插入 $L^k_{\text{in}}$; 否则对 $u$ 进行剪枝, 且不再继续遍历从 $u$ 开始的任何边。用同样的方法可以获得 $L^k_{\text{out}}$。

图 2 展示了一个有向图的 2-Hop 距离标签示例。图 2(a)是一个包含 7 个节点和 11 条边的有向图 $G$, 图 2(b)是图 $G$ 按节点度数递减顺序执行 PLL 算法生成的 2-Hop 距离标签。节点 $a$ 是第 3 个执行剪枝的 BFS 算法的节点, 初始化 $L^3 \leftarrow L^2$, $L^2_{\text{out}}=\{(c, 1), (d, 1)\}$, 从 $a$ 开始沿正向边方向执行剪枝的 BFS 获得 $L^3_{\text{in}}$, 首先访问 $a$, $d[a]$=0, 将($a$, 0)插入 $L^3_{\text{in}}(a)$; 接着访问 $b$, $d[b]$=1, 将($a$, 1)插入 $L^3_{\text{in}}(b)$; 然后访问 $c$ 和 $d$, 由于 distQuery($a$, $c$, $L^2$)=min{1+0}=1=$d[c]$, distQuery($a$, $d$, $L^2$) =min{1+1, 1+0}=$d[d]$, 故对节点 $c$ 和 $d$ 剪枝, $L^3_{\text{in}}(c)$和 $L^3_{\text{in}}(d)$不插入新的数据项且不再从 $c$ 和 $d$ 开始继续遍历。最后, 用同样的方法获得 $L^k_{\text{out}}$。由于对节点 $c$ 和 $d$ 进行剪枝, 所以($a$, 3)不会被插入 $L^3_{\text{in}}(g)$。




(a) 有向图$G$=($V$, $E$)

| | | |
|---|---|---|
| $L(a)$ | $L_{\text{out}}(a)$ | ($c$,1), ($d$,1), ($a$,0) |
| | $L_{\text{in}}(a)$ | ($a$,0) |
| $L(b)$ | $L_{\text{out}}(b)$ | ($c$,1), ($d$,1), ($b$,0) |
| | $L_{\text{in}}(b)$ | ($a$,1), ($b$,0) |
| $L(c)$ | $L_{\text{out}}(c)$ | ($c$,0) |
| | $L_{\text{in}}(c)$ | ($c$,0) |
| $L(d)$ | $L_{\text{out}}(d)$ | ($d$,0) |
| | $L_{\text{in}}(d)$ | ($c$,1), ($d$,0) |
| $L(e)$ | $L_{\text{out}}(e)$ | ($e$,0) |
| | $L_{\text{in}}(e)$ | ($c$,1), ($d$,1), ($e$,0) |
| $L(f)$ | $L_{\text{out}}(f)$ | ($f$,0) |
| | $L_{\text{in}}(f)$ | ($c$,1), ($d$,2), ($e$,1), ($f$,0) |
| $L(g)$ | $L_{\text{out}}(g)$ | ($g$,0) |
| | $L_{\text{in}}(g)$ | ($c$,2), ($d$,3), ($e$,2), ($f$,1), ($g$,0) |

(b) 图$G$的2-Hop距离标签

**图 2 图 $G$ 的 2-Hop 距离标签**

**Figure 2 2-Hop label of graph $G$**

### 4.1.2 支持最短路径查询的 2-Hop 标签

2-Hop 距离标签可以有效地支持图上最短距离查询并快速返回结果。虽然基本的 2-Hop 标签的构造简单, 但如果直接应用于最短路径查询则需要经过复杂的迭代和排序过程。为使 2-Hop 标签能更好地支持有向图的最短路径查询, 我们对 2-Hop 距离标签进行调整, 在 Akiba 等人[13]提出的剪枝地标标签算法基础上修改得到如算法 2 所示的 PLL 算法, 执行 PLL 算法获得支持有向图最短路径查询的 2-Hop 标签集 $L$。

PLL 算法首先初始化一个空标签字典 $L^0$, 然后将图 $G$ 的节点按度数递减顺序排序, 依次对每个节点执行剪枝的 BFS 算法 PrunedBFS, 最终输出支持最短路径查询的 2-Hop 标签 $L=\{L_{\text{out}}, L_{\text{in}}\}$。PrunedBFS 如算法 1 所示, 从节点 $v_k$ 分别沿正向边和反向边方向对图 $G$ 执行剪枝的 BFS, 在 $L^{k-1}$ 基础上计算标签集 $L^k$。在标签集 $L$ 中, 对于每一项($u$, $d_{vu}$, $w$)$\in L_{\text{out}}(v)$, $w$ 是从 $u$ 出发进行反向 BFS 时 $v$ 的前驱节点, 是从 $v$ 到 $u$ 最短路径上 $v$ 的下一个节点; 对于($u$, $d_{uv}$, $w$)$\in L_{\text{in}}(v)$, $w$ 是从 $u$ 出发进行正向 BFS 过程中 $v$ 的前驱节点, 即从 $u$ 到 $v$ 的最短路径上 $v$ 的上一个节点; 当且仅当元组中距离值为 0 时 $w$ 置为 null。

**算法 1.** $L^k \leftarrow$ PrunedBFS($G$, $v_k$, $L^{k-1}$)

输入: 有向图 $G=(V, E)$, 节点 $v_k$, 标签集 $L^{k-1}$

输出: 标签集 $L^k=\{L^k_{\text{out}}, L^k_{\text{in}}\}$

1: Initialize a queue $Q$ with one element $v_k$;
2: Initialize $L^k \leftarrow L^{k-1}$;
3: Set $d[v_k] \leftarrow 0$, $w[v_k] \leftarrow$null;
4: **FOR** $v \in V \setminus \{v_k\}$ **DO**
5: Set $d[v] \leftarrow \infty$, $w[v] \leftarrow$null;
6: **END FOR**
7: **WHILE** $Q$ is not empty **DO**
8: Dequeue $u$ from $Q$;
9: **IF** distQuery($v_k$, $u$, $L^{k-1}$) $\leqslant d[u]$ **THEN**
10: **continue**;
11: **END IF**
12: Add ($v_k$, $d[u]$, $w[u]$) into $L^k_{\text{in}}(u)$;
13: **FOR** $<u, v_i> \in E$ s.t. $d[v_i]=\infty$ **DO**
14: $d[v_i] \leftarrow d[u]+1$, $w[v_i] \leftarrow u$;
15: Enqueue $v_i$ to $Q$;
16: **END FOR**
17: **END WHILE**
18: Enqueue $v_k$ to $Q$;
19: Replace (i) distQuery($v_k$, $u$, $L^{k-1}$) with distQuery ($u$, $v_k$, $L^{k-1}$), (ii) $L^k_{\text{in}}(u)$ with $L^k_{\text{out}}(u)$ and (iii) $<u, v_i>$ with $<v_i, u>$, and then repeat the procedure in line 3-17;
20: Output $L^k=\{L^k_{\text{out}}, L^k_{\text{in}}\}$.

**算法 2.** $L \leftarrow$ PLL($G$)

输入: 有向图 $G=(V, E)$

输出: 标签集 $L$

1: Initialize an empty dictionary $L^0$;
2: Sort the nodes of $G$ in descending order of degree;

3: **FOR** $k$ =1 to $|V|$ **DO**
4: $L^k \leftarrow$ PrunedBFS($G$, $v_k$, $L^{k-1}$);
5: **END FOR**
6: $L \leftarrow L^{|V|}$;
7: Output $L$.

对于任意两个可达节点 $s$ 和 $t$, 在支持最短路径查询的 2-Hop 标签集 $L$ 上使用以下公式计算和回答从 $s$ 到 $t$ 的最短距离查询 distQuery($s$, $t$, $L$):

$$\text{distQuery}(s,\ t,\ L)=\min\{d_{su}+d_{ut}\} \tag{2}$$

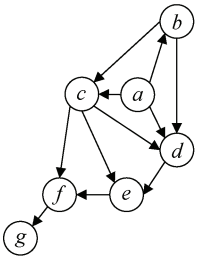
其中, $(u,\ d_{su},\ w_1)\in L_{\text{out}}(s)$且$(u,\ d_{ut},\ w_2)\in L_{\text{in}}(t)$。如果 $L_{\text{out}}(s)$和 $L_{\text{in}}(t)$没有共享的节点 $u$, 即 $s$ 到 $t$ 不可达, 则定义 $\text{distQuery}(s,t,L)=\infty$ 。

在 2-Hop 标签上进行最短路径查询的过程如下: 给定起点 $s$ 和终点 $t$, 获取 $L_{\text{out}}(s)$和 $L_{\text{in}}(t)$, 假如从 $s$ 到 $t$ 可达, 至少存在一个共享节点 $v$ 在从 $s$ 到 $t$ 的最短路径上, 其对应数据项$(v,\ d_{sv},\ w_1)\in L_{\text{out}}(s)$和$(v,\ d_{vt},\ w_2)\in L_{\text{in}}(t)$, 则从 $s$ 到 $t$ 最短距离为 $d_{st}=d_{sv}+d_{vt}$, 获取当前起点 $s$ 的下一个节点 $w_1$ 和终点 $t$ 的上一个节点 $w_2$。由于最短路径的子路径也是最短路径[11], 将 $w_1$ 和 $w_2$ 作为新的起点 $s$ 和终点 $t$, 计算下一对 $w_1$ 和 $w_2$。以此类推, 如果在某一轮循环中 $w_1$ 为 null, 则 $s$ 保持不变, 只更新 $t$ 为 $w_2$; 如果 $w_2$ 为 null, 则 $t$ 不变, 只更新 $s$ 为 $w_1$。直到 $w_1$ 和 $w_2$ 均为 null, 此时, 对应的起点、终点和共享节点三者重合, 路径搜索结束。

图 3 是一个支持图 2(a)所示有向图 $G$ 最短路径查询的 2-Hop 标签示例。若要在这个标签集上查询从 $b$ 到 $g$ 的最短路径, 首先检查 $L_{\text{out}}(b)$和 $L_{\text{in}}(g)$, 可以观察到 $L_{\text{out}}(b)$和 $L_{\text{in}}(g)$有共享节点 $c$ 和 $d$, 由公式(1)可知从 $b$ 到 $g$ 的最短距离 $d_{bg}=\min\{1+2,\ 1+3\}=3$, $c$ 在这条最短路径上, 对应项为$(c,\ 1,\ c)\in L_{\text{out}}(b)$和$(c,\ 2,\ f)\in L_{\text{in}}(g)$, 则起点 $b$ 的下一个节点和终点 $g$ 的上一个节点分别为 $c$ 和 $f$; 接着, 以 $c$ 和 $f$ 为新的起点和终点, 检查 $L_{\text{out}}(c)$和 $L_{\text{in}}(f)$, 共享节点只有 $c$, 仅能找出 $f$ 的上一个节点为 $c$; 将终点更新为 $c$, 检查 $L_{\text{out}}(c)$和 $L_{\text{in}}(c)$, 找出共享节点 $c$, 起点、终点和共享节点重合, 路径上不再有其他节点, 因此从 $b$ 到 $g$ 的最短路径为 $b\rightarrow c\rightarrow f\rightarrow g$。

### 4.2 改进的保序编码 OPE

由公式(2)可知, 在 2-Hop 标签上计算两节点间最短距离需要进行一次加法运算与值的比较。因此, 本文方案需要一种特殊的保序编码(Order- Preserving Encoding, OPE)算法, 在保护原始数据的同时, 使其在执行一次加法运算后仍能保留数值的顺序特征。本文采用 Liu 等人[9]提出的改进的保序编码算法 OPE。如算法 3 所示, 输入正整数密钥 $K_2$ 和整数 $d$, 生成随机值 $r\leftarrow^{\$}(0, K_2/2)$, OPE 将整数 $d$ 编码为 $D\leftarrow K_2\cdot d+r$。对于任意 4 个整数值 $d_1$、$d_2$、$d_3$ 和 $d_4$, 对应的保序编码分别为 $D_i\leftarrow\text{OPE}(K_2, d_i)$, $i\in[1, 4]$。如果 $d_1+d_2>d_3+d_4$, 那么 $D_1+D_2>D_3+D_4$ 成立。




(a) 有向图$G=(V, E)$

| | | |
|---|---|---|
| $L(a)$ | $L_{\text{out}}(a)$ | ($c$,1,$c$), ($d$,1,$d$), ($a$,0,null) |
| | $L_{\text{in}}(a)$ | ($a$,0,null) |
| $L(b)$ | $L_{\text{out}}(b)$ | ($c$,1,$c$), ($d$,1,$d$), ($b$,0,null) |
| | $L_{\text{in}}(b)$ | ($a$,1,$a$), ($b$,0,null) |
| $L(c)$ | $L_{\text{out}}(c)$ | ($c$,0,null) |
| | $L_{\text{in}}(c)$ | ($c$,0,null) |
| $L(d)$ | $L_{\text{out}}(d)$ | ($d$,0,null) |
| | $L_{\text{in}}(d)$ | ($c$,1,$c$), ($d$,0,null) |
| $L(e)$ | $L_{\text{out}}(e)$ | ($e$,0,null) |
| | $L_{\text{in}}(e)$ | ($c$,1,$c$), ($d$,1,$d$), ($e$,0,null) |
| $L(f)$ | $L_{\text{out}}(f)$ | ($f$,0,null) |
| | $L_{\text{in}}(f)$ | ($c$,1,$c$), ($d$,2,$e$), ($e$,1,$e$), ($f$,0,null) |
| $L(g)$ | $L_{\text{out}}(g)$ | ($g$,0,null) |
| | $L_{\text{in}}(g)$ | ($c$,2,$f$ ), ($d$,3,$f$ ), (e,2,$f$), ($f$,1,$f$), ($g$,0,null) |

(b) 支持最短路径查询的2-Hop标签

**图 3 支持最短路径查询的 2-Hop 标签**

**Figure 3 2-Hop labels supporting shortest path query**

**正确性.** 对于任意整数密钥 $K$ 和 4 个整数值 $d_1$、$d_2$、$d_3$ 和 $d_4$, 如果 $d_1+d_2>d_3+d_4$, 那么,

$$\begin{aligned}&D_1+D_2-(D_3+D_4)\\&=K_2\cdot d_1+r_1+K_2\cdot d_2+r_2-(K_2\cdot d_3+r_3+K_2\cdot d_4+r_4)\\&=K_2\cdot(d_1+d_2-d_3-d_4)+(r_1+r_2-r_3-r_4)\\&\geqslant K_2\cdot 1+(r_1+r_2-r_3-r_4)\\&>K_2+(-K_2)=0\end{aligned}$$

所以, $D_1+D_2>D_3+D_4$ 成立。

**算法 3.** $D\leftarrow\text{OPE}(K_2, d)$

输入: 密钥 $K_2$, 整数 $d$
输出: 保序编码值 $D$

1: $r\leftarrow^{\$}(0, K_2/2)$;
2: $D\leftarrow K_2\cdot d+r$;
3: Output $D$.

## 5 结构化加密图的最短路径查询方案

本节提出支持有向图上任意两节点间最短路径查询的结构化加密图方案 Π = (KeyGen, GraphEnc, Token, PathQuery, Decrypt), 并详细介绍组成该方案的 5 个算法。

### 5.1 密钥生成算法 KeyGen

密钥生成算法 KeyGen 如算法 4 所示, 给定安全参数 $\lambda$, 数据拥有者生成包含三个密钥的密钥集 $K=\{K_1, K_2, K_3\}$, 其中 $K_1$ 是一个 $\lambda$ 位的随机字符串, 用于生成各节点的标签索引入口; $K_2$ 和 $K_3$ 分别是保序编码算法 OPE 和对称密钥加密 SKE 的密钥, 用于图数据的加密和令牌的生成。

**算法 4.** $K \leftarrow \text{KeyGen}(1^{\lambda})$

输入: 安全参数 $\lambda$

输出: 密钥集 $K$

1: $K_1 \leftarrow^{\$} \{0,1\}^{\lambda}$;

2: $K_2 \leftarrow^{\$} \mathbb{Z}_{2^{\lambda}}$;

3: $K_2 \leftarrow \text{SKE.Gen}(1^{\lambda})$;

4: Output $K=\{K_1, K_2, K_3\}$.

## 5.2 加密算法 GraphEnc

图的加密算法 GraphEnc 如算法 5 所示。数据拥有者首先使用算法 2 所示的 PLL 算法为有向图 $G$ 生成支持最短路径查询的 2-Hop 标签 $L=\{L_{\text{out}}, L_{\text{in}}\}$, 然后, 为每个节点 $v$ 对应的标签集 $L_{\text{out}}(v)$和$L_{\text{in}}(v)$生成一对索引入口$(I_v^+, K_v^+)$和$(I_v^-, K_v^-)$, 并对标签项进行加密和混淆, 得到加密的标签字典 $EL=\{EL_{\text{out}}, EL_{\text{in}}\}$。

对于$(u, d_{vu}, w)\in L_{\text{out}}(v)$, 数据拥有者生成键 $I_{vu}^+$, 然后计算 $u$ 的伪随机函数值, 用 OPE 对距离值 $d_{vu}$ 编码。但在对从 $v$ 到 $u$ 的最短路径第一个节点 $w$ 进行加密时, 不直接使用 SKE.Enc 加密节点 $w$, 而是加密相应的边<$v$, $w$>, <$v$, $w$>揭示从 $v$ 到 $u$ 的最短路径上的第一条边。此外, 在数据项中保存节点 $w$ 对应的 out 标签的索引入口值 $\tau_w^+=(I_w^+, K_w^+)$, 该值将用于在最短路径查询时作为新令牌的一部分查找最短路径上的下一对边。最后使用伪随机函数 $g(K_v^+, ctr)$ 将数据项进行混淆后插入字典。数据拥有者使用类似的方法加密 $L_{\text{in}}(v)$中的每一项。同理, 对于$(u, d_{uv}, w)\in L_{\text{in}}(v)$, 加密<$w$, $v$>而不是 $w$, <$w$, $v$>揭示从 $u$ 到 $v$ 的最短路径上的最后一条边。加密完成后, 数据拥有者得到加密的图标签字典 $EL=\{EL_{\text{out}}, EL_{\text{in}}\}$发送给云服务器。

**算法 5.** $EL \leftarrow \text{GraphEnc}(K, G)$

输入: 密钥集 $K$, 图 $G=(V, E)$

输出: 加密标签 $EL$

1: Compute $L \leftarrow \text{PLL}(G)$ and parse $L$ as $L_{\text{out}}$ and $L_{\text{in}}$;

2: Parse $K$ as $K_1, K_2, K_3$;

3: Initialize two dictionaries $EL_{\text{out}}$ and $EL_{\text{in}}$;

4: **FOR** $v\in V$ **DO**

5: Calculate $I_v^+ := h(K_1, v||0)$, $K_v^+ := h(K_1, v||1)$;

6: Calculate $I_v^- := h(K_1, v||2)$, $K_v^- := h(K_1, v||3)$;

7: **END FOR**

8: **FOR** $v\in V$ **DO**

9: Set a counter $ctr = 0$;

10: **FOR** $(u, d_{vu}, w)\in L_{\text{out}}(v)$ **DO**

11: Calculate $I_{vu}^+ := h(I_v^+, ctr)$;

12: Compute $H_u := h(K_1, u)$;

13: Compute $D_{vu} \leftarrow \text{OPE}(K_2, d_{vu})$;

14: **IF** $w \neq$ null **THEN**

15: Compute $c_{vw} \leftarrow \text{SKE.Enc}(K_3, \langle v,w\rangle)$;

16: Set $\tau_w^+ := (I_w^+, K_w^+)$;

17: **ELSE**

18: Pad $c_{vw}$ and $\tau_w^+$ with random strings;

19: **END IF**

20: Set $EL_{\text{out}}[I_{vu}^+] := (H_u, D_{vu}, c_{vw}, \tau_w^+) \oplus g(K_v^+, ctr)$;

21: Set $ctr := ctr+1$;

22: **END FOR**

23: Set $ctr = 0$;

24: Replace (i) subscript out with in, subscript $vu$ with $uv$ and superscript + with −, (ii) $c_{vw}$ with $c_{wv}$, (iii) <$v$,$w$> with <$w$,$v$> and then repeat the procedure in line 10-22;

25: **END FOR**

26: Output $EL=\{EL_{\text{out}}, EL_{\text{in}}\}$.

图 4 展示了图 3(b)中有向图 $G$ 的支持最短路径查询的 2-Hop 标签的加密结果, 假设距离值为整数 $d$, $D(d)$表示 $d$ 的保序编码值, *表示随机字符串。

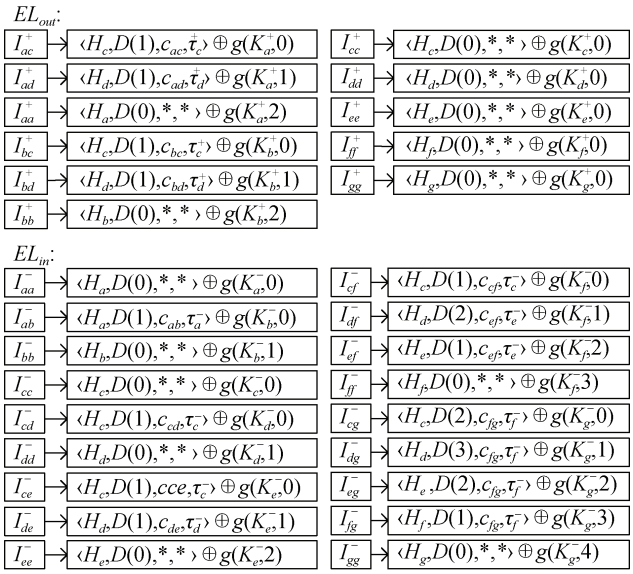



**图 4 加密的 2-Hop 标签**

**Figure 4 Encrypted 2-Hop labels**

## 5.3 令牌生成算法 Token

令牌生成算法 Token 如算法 6 所示, 数据使用者向数据拥有者发送查询请求 $q=(s, t)$, 查询从起点 $s$ 出发到终点 $t$ 的最短路径; 数据拥有者计算令牌 $\tau_{st}$ 发送给数据使用者; 数据使用者将令牌 $\tau_{st}$ 发送给云服务器进行检索。

**算法 6.** $\tau_{st}$ ←Token($K$, $q$)

输入: 密钥集 $K$, 查询 $q$=($s$, $t$)

输出: 令牌 $\tau_{st}$

Parse $K$ as $K_1$, $K_2$, $K_3$;
Calculate $I_s^+ := h(K_1, s||0)$, $K_s^+ := h(K_1, s||1)$;
Calculate $I_t^- := h(K_1, t||2)$, $K_t^- := h(K_1, t||3)$;
Output $\tau_{st} := (I_s^+, K_s^+, I_t^-, K_t^-)$.

## 5.4 路径查询算法 PathQuery

路径查询算法 PathQuery 由云服务器运行以计算最短路径, 使用两个子算法: 标签搜索算法 LabelSearch 和边搜索算法 NextEdgeSearch。

标签搜索算法 LabelSearch 以加密的图标签集 $EL$ 和令牌 $\tau_{st}$ 为输入, 获取 $EL_{\text{out}}$ 中与 $s$ 关联的标签项和 $EL_{\text{in}}$ 中与 $t$ 关联的标签项分别插入 $L_s$ 和 $L_t$ 并返回, 如算法 7 所示。

边搜索算法NextEdgeSearch如算法8所示, 输入LabelSearch 的结果 $L_s$ 和 $L_t$, 云服务器解析 $L_s$ 和 $L_t$ 中各数据项进行比较, 如果存在 $H_u$=$H_v$, 则表明 $L_{\text{out}}(s)$ 和 $L_{\text{in}}(t)$具有共享节点, 从 $s$ 到 $t$ 是可达的。利用 Liu 等人[9]提出的 OPE 算法特性, 云服务器通过比较找出距离之和最短的一对标签项, 从而得到从 $s$ 到 $t$ 最短路径上第一条边的加密值 $c_1$ 和最后一条边的加密值 $c_2$, 以及 $s$ 的下一个节点和 $t$ 的上一个节点的索引入口, 用于查找最短路径上的下一对边。

**算法 7.** ($L_s$, $L_t$) ←LabelSearch($EL$, $\tau_{st}$)

输入: 加密标签 $EL$, 令牌 $\tau_{st}$

输出: 字典 $L_s$, $L_t$

Parse $\tau_{st}$ as $(I_s^+, K_s^+, I_t^-, K_t^-)$;
Initialize two dictionaries $L_s$ and $L_t$;
Set a counter $ctr$ = 0;
Calculate $I_{su}^+ := h(I_s^+, ctr)$;
**FOR** $EL_{\text{out}}[I_{su}^+] \neq \perp$ **DO**
    Compute $\Phi_{su} := EL_{\text{out}}[I_{su}^+] \oplus g(K_s^+, ctr)$;
    Parse $\Phi_{su}$ as $(H_u, D_{su}, c_{sw}, \tau_w^+)$;
    Set $L_s[H_u] := (D_{su}, c_{sw}, \tau_w^+)$;
    Set $ctr$ = $ctr$+1;
    Calculate $I_{su}^+ := h(I_s^+, ctr)$;
**END FOR**
Set $ctr$ = 0;
Calculate $I_{vt}^- := h(I_t^-, ctr)$;
**FOR** $EL_{in}[I_{vt}^-] \neq \perp$ **DO**
    Compute $\Phi_{vt} := EL_{\text{in}}[I_{vt}^-] \oplus g(K_t^-, ctr)$;
    Parse $\Phi_{vt}$ as $(H_v, D_{vt}, c_{wt}, \tau_w^-)$;
    Set $L_t[H_v] := (D_{vt}, c_{wt}, \tau_w^-)$;
    Set $ctr$ = $ctr$+1;
    Calculate $I_{vt}^- := h(I_t^-, ctr)$;
**END FOR**
Return ($L_s$, $L_t$).

**算法 8.** ($c_1$, $c_2$, $\tau'_{st}$) ← NextEdgeSearch ($L_s$, $L_t$)

输入: 字典 $L_s$, $L_t$

输出: 加密边 $c_1$, $c_2$, 新令牌 $\tau'_{st}$

Initialize $dist$← ∞;
**FOR** $L_s[H_u] \in L_s$, $L_t[H_v] \in L_t$ **DO**
    Parse $L_s[H_u]$ as $(D_{su}, c_{sw_i}, \tau_{w_i}^+)$;
    Parse $L_t[H_v]$ as $(D_{vt}, c_{w_jt}, \tau_{w_j}^-)$;
    **IF** $H_u$=$H_v$ **THEN**
        **IF** $dist > D_{su}+D_{vt}$ **THEN**
            Compute $dist$←$D_{su}$+ $D_{vt}$;
            Set $c_1$← $c_{sw_i}$, $c_2$← $c_{w_jt}$;
            Set $\tau'_{st}$:=( $\tau_{w_i}^+$, $\tau_{w_j}^-$);
        **END IF**
    **END IF**
**END FOR**
Return ($c_1$, $c_2$, $\tau'_{st}$).

路径查询算法 PathQuery 如算法 9 所示。云服务器收到令牌 $\tau_{st}$ 后, 运行 LabelSearch 算法获取与 $EL_{\text{out}}(s)$和 $EL_{\text{in}}(t)$的标签项插入 $L_s$ 和 $L_t$。检查 $L_s$ 和 $L_t$, 如果 $L_s$ 为空或 $L_t$ 为空, 说明节点 $s$ 或 $t$ 不存在, 则 $R$ 为空。当 $L_s$ 和 $L_t$ 均不为空时说明从 $s$ 到 $t$ 可达, 云服务器执行 NextEdgeSearch 算法, 返回从 $s$ 到 $t$ 的最短路径上的第一条边的加密值 $c_1$、最后一条边的加密值 $c_2$ 和用于查找下一对边的新令牌 $\tau'_{st}$。使用 $\tau'_{st}$ 运行 LabelSearch 算法获取 $L'_s$ 和 $L'_t$, 根据 $L'_s$ 和 $L'_t$ 的情况决定是否添加新的加密边到 $R$ 或 $R'$中和更新 $L_s$ 与 $L_t$; 当 $L'_s$ 与 $L'_t$ 均为空, 表明已找出路径上所有边, 循环结束。最后, 反转 $R'$中元素的顺序并依次添加到 $R$, $R$ 即是所求的最短路径上节点的密文。

**算法 9.** $R$←PathQuery($EL$, $\tau_{st}$)

输入: 加密标签 $EL$, 令牌 $\tau_{st}$

输出: 加密结果 $R$

Initialize two sequential lists $R$ and $R'$;
Compute ($L_s$, $L_t$) ←LabelSearch($EL$, $\tau_{st}$);
**IF** $L_s$=⊥ or $L_t$=⊥ **THEN**
    Set $R$ =⊥;
**END IF**
**FOR** $L_s \neq \perp$, $L_t \neq \perp$ **DO**
    ($c_1$, $c_2$, $\tau'_{st}$)← NextEdgeSearch ($L_s$, $L_t$)
    ($L'_s$, $L'_t$)←LabelSearch($EL$, $\tau'_{st}$)
    **IF** $L'_s \neq \perp$, $L'_t \neq \perp$ **THEN**

10: Set $L_s := L'_s$, $L_t := L'_t$;
11: Append $c_1$ to $R$, append $c_2$ to $R'$;
12: **ELSE IF** $L'_s \neq \perp$ , $L'_t = \perp$ **THEN**
13: Set $L_s := L'_s$;
14: Append $c_1$ to $R$;
15: **ELSE IF** $L'_s = \perp$, $L'_t \neq \perp$ **THEN**
16: Set $L_t := L'_t$;
17: Append $c_2$ to $R'$
18: **ELSE**
19: **break;**
20: **END IF**
21: **END FOR**
22: Reverse the order of elements in $R'$ and then append the elements in $R'$ to $R$ in sequence;
23: Output $R$.

图 5 示例了在图 4 所示的加密图标签集上查询从节点 $b$ 到 $g$ 的最短路径的过程。如图 5(a)所示, 云服务器根据 $\tau_{bg}$ 获取 $EL_{\text{out}}(b)$和 $EL_{\text{in}}(g)$中数据项分别插入 $L_s$ 和 $L_t$, 发现两个共享节点, 当共享节点伪随机标识符为 $H_c$ 时距离最短, 从而找到从 $b$ 到 $g$ 最短路径上加密的第一条边 $c_{bc}$ 和最后一条边 $c_{fg}$, 新令牌为 $\tau'_{st}=(\tau_c^+, \tau_f^-)$。接着, 如图 5(b)所示, 使用 $\tau'_{st}=(\tau_c^+, \tau_f^-)$求解从 $c$ 到 $f$ 最短路径上的边, 获得 $EL_{\text{out}}(c)$和 $EL_{\text{in}}(f)$的数据项, 两者均不为空, 所以更新 $L_s$ 和 $L_t$, 计算获得 $c_{fg}$ 的上一条边 $c_{cf}$, 新令牌为 $\tau'_{st}:=(*,\tau_c^-)$。如图 5(c)所示, 由于以*作为索引入口值在 $EL_{\text{out}}$ 中找不到关联数据项, 因此不更新 $L_s$。本轮获得新的令牌 $\tau'_{st}:=(*, *)$, 但凭此令牌运行 LabelSeach 结果为空, 路径上不再有其他边, 最终得到加密路径结果 $R=\{c_{bc}, c_{cf}, c_{fg}\}$。

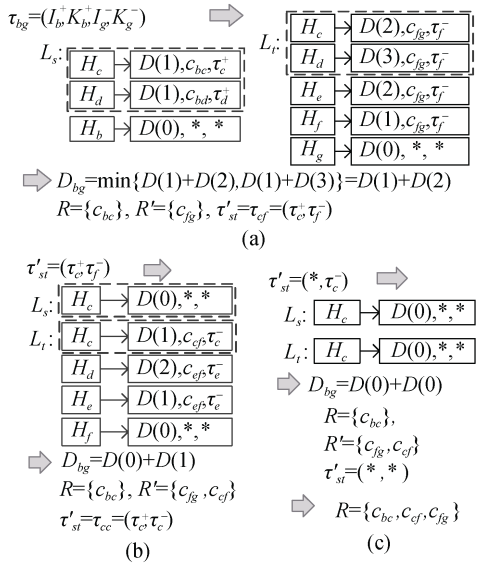



**图 5 查询过程示例**

**Figure 5 Example of query process**

### 5.5 解密算法 Decrypt

解密算法 Decrypt 如算法 10 所示。数据使用者使用从数据拥有者处得到的密钥 $K_3$, 依次解密 $R$ 中的加密数据项 $c_1, c_2, \ldots, c_{|R|}$, 得到路径明文 $p$。

**算法 10.** $p \leftarrow \text{Decrypt}(K_3, R)$

输入: 密钥 $K_3$, 加密结果 $R$
输出: 路径 $p$
1: Parse $R$ as $(c_1, c_2, \ldots, c_{|R|})$;
2: **FOR** $i = 1$ to $|R|$ **DO**
3: $m_i \leftarrow \text{SKE.Dec}(K_3, c_i)$;
4: Append $m_i$ into $p$;
5: **END FOR**
6: Output $p$.

## 6 安全性分析

本节证明本文提出的支持最短路径查询的结构化加密图方案 Π = (KeyGen, GraphEnc, Token, PathQuery, Decrypt) 在 CQA2-安全模型下是自适应安全的, 即结构化加密图方案 Π 对于自适应选择查询攻击是 $(\mathcal{L}_1, \mathcal{L}_2)$-安全的。

**定理 1.**如果本文提出的结构化加密图方案 Π = (KeyGen, GraphEnc, Token, PathQuery, Decrypt)满足泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$, 那么方案 Π 对自适应选择查询攻击(CQA2)是 $(\mathcal{L}_1, \mathcal{L}_2)$-安全的。

**证明.** 证明的主要思路是构造一个模拟器 $\mathcal{S}$。$\mathcal{S}$ 根据给定的泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 模拟一个虚拟的加密图标签 $EL^* = \{EL^*_{\text{out}}, EL^*_{\text{in}}\}$ 和查询序列 $\boldsymbol{q}^*$。如果任意的概率多项式时间敌手 $\mathcal{A}$ 不能区分两个实验 $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ 和 $\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$, 则可以认为结构化加密图方案 Π 对自适应选择查询攻击是 $(\mathcal{L}_1, \mathcal{L}_2)$-安全的。

**模拟** $EL^*$: 给定泄露函数 $\mathcal{L}_1$, 模拟器 $\mathcal{S}$ 根据 $\mathcal{L}_1$ 生成基于图 $G$ 标签 $L_{\text{out}}$ 的虚拟加密标签 $EL^*_{\text{out}}$。首先, $\mathcal{S}$ 生成模拟密钥 $K^* = \{K_1^*, K_2^*, K_3^*\}$。假设节点集为 $V=\{v_1, \ldots, v_n\}$, $\mathcal{S}$ 为每个节点 $v_i$ 随机选取一个正整数 $\omega_i$ 使得 $\sum_{i=1}^{n} \omega_i = |L_{\text{out}}|$。接着, 均匀选取不重复的两个随机值 $\alpha_i \leftarrow \{0,1\}^\lambda, \beta_i \leftarrow \{0,1\}^\lambda$。对于 $0 \leqslant j < \omega_i$, $\mathcal{S}$ 执行以下步骤模拟生成 $EL^*_{\text{out}}(v_i)$:

(1) $\mathcal{S}$ 计算 $\alpha_{ij} = h(\alpha_i, j)$, $\beta_{ij} = g(\beta_i, j)$, $h$ 和 $g$ 是两个伪随机函数;

(2)假设 $u$ 是 $L_{\text{out}}(v_i)$中第 $j$ 个标签项的共享节点, $\mathcal{S}$ 使用 $H^*_u = h(K_1^*, u)$ 计算虚拟的伪随机标识符。随机选取非负整数 $d^*_{ij}$ 和节点 $v_k$ 作为虚拟的距离值和下

一节点, 然后使用 OPE 编码 $d_{ij}^*$ 得到 $D_{ij}^*$, 使用 SKE 加密<$v_i$, $v_k$>得到 $c_{ik}^*$, 计算 $\tau_k^* = (\alpha_k, \beta_k)$;

(3) $\mathcal{S}$ 置 $EL_{\text{out}}^*[\alpha_{ij}] := \beta_{ij} \oplus (H_u^*, D_{ij}^*, c_{ik}^*, \tau_k^*)$。

$\mathcal{S}$ 用同样的方法生成虚拟的标签集 $EL_{\text{in}}^*$, 最后获得虚拟的加密 2-Hop 标签 $EL^* = \{EL_{\text{out}}^*, EL_{\text{in}}^*\}$。

**模拟 $\boldsymbol{q}^*$**: 给定泄露函数 $\mathcal{L}_2$, 假设敌手 $\mathcal{A}$ 生成一组查询序列 $\boldsymbol{q}^*$={($s_1$, $t_1$), …, ($s_n$, $t_n$)}, 对于任意一个查询 $q_i^* = (s_i, t_i) \in \boldsymbol{q}^*$, 根据给定的 $\mathcal{L}_2$, $\mathcal{S}$ 按以下过程模拟虚拟令牌 $\tau_i^*$:

(1) $\mathcal{S}$ 检查节点 $s_i$ 和 $t_i$ 是否与先前的查询重复。如果 $s_i$ 已在查询中出现过, $I_{s_i}^{*+}$ 和 $K_{s_i}^{*+}$ 取先前的值;

(2)如果 $s_i$ 未在查询中出现过, $\mathcal{S}$ 选择一对未使用过的 $\alpha_i$ 和 $\beta_i$ 给 $I_{s_i}^{*+}$ 和 $K_{s_i}^{*+}$ 赋值, 并记录 $s_i$、$\alpha_i$ 和 $\beta_i$ 的对应关系。

(3) $\mathcal{S}$ 用同样的方法为节点 $t_i$ 选取 $I_{t_i}^{*-}$ 和 $K_{t_i}^{*-}$, 最后得到模拟的令牌 $\tau_i^* = (I_{s_i}^{*+}, K_{s_i}^{*+}, I_{t_i}^{*-}, K_{t_i}^{*-})$。

$\tau_i^*$ 是虚拟令牌序列 $T_{st}^*$ 的一个元素, 由于查询序列 $\boldsymbol{q}^*$是多项式的, 模拟器 $\mathcal{S}$ 可以通过 $T_{st}^*$ 在多项式时间内模拟令牌序列。

因为 $g$ 和 $h$ 是伪随机函数, SKE 在选择明文攻击下具有伪随机的密文, 虚拟的加密标签 $EL^*$ 和模拟的令牌序列 $T_{st}^*$ 在真实实验 $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ 和理想实验 $\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ 中不可区分, 方案满足:

$$|\Pr[\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)=1] - \Pr[\mathbf{Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)=1]| \leqslant \text{negl}(\lambda)$$

因此, 方案 Π 对自适应选择查询攻击是 $(\mathcal{L}_1, \mathcal{L}_2)$-安全的。

## 7 性能分析

本节对提出的支持最短路径查询的结构化加密图方案进行性能分析。实验环境是配置为 Intel(R) Core(TM) i7-10750H CPU 2.30 GHz、16 GB 内存的 PC, 64-bit Win 10 的操作系统。实验利用 VMware 虚拟机加载开源项目 OpenStack 进行性能测试, 使用 python 语言编程实现算法。实验数据集选用斯坦福网络分析平台(SNAP)的 email-Eu-core 数据集①进行评估。该网络由一家大型欧洲研究机构的电子邮件数据生成, 包含 1005 个节点和 25571 条有向边, 节点代表用户, 从节点 $u$ 到 $v$ 的有向边表示用户 $u$ 向用户 $v$ 发出过邮件。

本节将从存储空间、各阶段的时间开销和通信开销对提出的方案进行分析与评估。

本文方案选择安全参数 $\lambda$=256, 使用 HMAC-SHA256 计算伪随机函数, 以 CBC 模式的 AES-256 算法实现 SKE。我们将同时测试 SecGDB[6]、文献 PGAS[8]、文献[27]中 Ramezanian 等人提出的方案和 GES[29]用于性能比较。

### 7.1 复杂度分析

本方案中, 数据拥有者仅需存储 3 个密钥, 长 3$\lambda$。加密图在服务器端的存储空间复杂度如表 2 所示, 对于图 $G$=($V$, $E$), 节点数和边数分别记为 $n$ 和 $m$, $l$ 是最短路径上的边数, $k$ 表示与节点 $v$ 相关联的标签集 $L(v)$可能包含的最大项数。

**表 2 方案对比**

**Table 2 Scheme comparison**

| 方案 | 查询功能 | 存储空间复杂度 | 查询时间复杂度 | 通信复杂度 |
|---|---|---|---|---|
| SecGDB[6] | 最短距离和最短路径查询 | $O(n+m)$ | $O(n\log n+m)$ | $O(l)$ |
| PGAS[8] | 精确约束最短距离查询 | $O(n\sqrt{m})$ | $O(k)$ | $O(1)$ |
| 文献[27] | 最短路径查询 | $O(n^2)$ | $O(n^2 l)$ | $O(l^2 n)$ |
| GES[29] | 最短路径查询 | $O(n^2)$ | $O(l)$ | $O(l)$ |
| 本文方案 | 最短路径查询 | $O(n\sqrt{m})$ | $O(lk)$ | $O(l)$ |

SecGDB[6]方案存储一个加密的邻接表, 空间复杂度为 $O(n+m)$; 在最坏情况下, 任意节点对间存在一条最短路径, GES[29]方案需存储一个 $O(n^2)$大小的字典; 文献[27]的方案存储一个扩展的传递闭包矩阵和一个密钥矩阵, 空间复杂度 $O(n^2)$; 对于 PGAS[8]和本文的方案, 加密的图标签存储空间复杂度依赖于标签大小, 2-Hop 标签的大小可以实现 $O(n\sqrt{m})$ 的存储开销, 对于真实数据集中的图, 边数通常最多为节点数的二十到三十倍, 在实际构造中其结构大小往往远小于 $O(n^2)$, 可有效节约存储空间。

SecGDB 和 GES 令牌均由若干个伪随机函数值组成, PGAS 和本文方案的令牌都包括 4 个伪随机函数值, PGAS 另有 1 个公钥加密的约束值, 因此, 以上方案的令牌生成算法时间和空间复杂度均为 $O(1)$。在查询过程中, 云服务器根据数据使用者发送的令牌进行检索。如表 2 所示, SecGDB 方案的检索最短路径的方法是在加密图上执行 Dijkstra 算法, 时间复

① https://snap.stanford.edu/data/email-Eu-core.html.

杂度为 $O(n\log n+m)$, 但其中的计算和比较操作使用 Paillier 同态加密和混淆电路完成。PGAS 方案执行精确约束最短距离查询时间复杂度 $O(n)$, 应用 PCTD 构建比较协议。Ramezanian 等人[27]提出的方案是由客户端和云服务器在加密的密钥矩阵上执行一个私有信息检索(PIR)协议, 云服务器需进行 $l$ 次$(n^2+n)$次模指数运算; GES 通过存储的最短路径矩阵循环求解最短路径上的节点, 时间复杂度只与最短路径上边数 $l$ 有关。本文结合 2-Hop 标签和这种递归思想, 虽然复杂度同时与路径长度 $l$ 和各节点的标签大小相关, 但使用 OPE 代替基于 PCTD 的比较协议完成比较操作, 计算开销的增加是可接受的。

表 2 也对查询过程中各方案的用户和服务器间的通信开销进行比较。在 SecGDB[6]、文献 PGAS[8]、GES[29]和本文方案的查询过程中, 用户获取令牌发送给服务器, 服务器执行查询算法返回加密结果, 因为在这 4 个方案中令牌的计算和查询算法的运行不涉及用户和服务器之间的交互, 所以用户和服务器之间的通信包括令牌的发送和加密结果的传输两部分。在发送令牌的过程中, 由于 4 个方案的令牌均由常数个伪随机函数值或加密值组成, 其通信复杂度均为 $O(1)$; 在云服务器向用户返回加密结果的部分, PGAS 方案返回一个加密距离值, 通信复杂度为 $O(1)$, GES 和本文方案均返回最短路径, 通信复杂度与最短路径上的边数有关, 为 $O(l)$。而 Ramezanian 等人[27]的方案在计算令牌和查询过程中需要在客户端和云服务器之间执行 $l$ 次 PIR 协议, 通信复杂度达 $O(l^2n)$。此外, 尽管 SecGDB 和 PGAS 方案中用户和云服务器之间的通信同样只有两次, 但 SecGDB 需要一个云服务器和一个代理服务器共同完成值的比较和最短路径的计算, PGAS 方案需要一个云存储服务器和若干云计算服务器协同完成比较和最小值的计算, 在服务器之间存在额外的通信, 而 GES 方案和本方案则由单个云服务器完成。

## 7.2 实验评估

本小节通过实验对方案各阶段算法的空间存储开销, 时间开销和通信开销情况进行分析。

### 7.2.1 存储开销分析

我们首先分析方案的各阶段算法的存储空间开销。在本文方案中, 数据拥有者保存生成的 3 个密钥, 共 96 字节。

图标签索引的构造和加密是一次性计算, 主要包括两个方面: 构造支持最短路径查询的索引和对索引加密, 后者是本文关注的重点。本次实验从实验数据集中选取具有不同节点数的子图, 构造符合要求的明文索引, 将其作为输入, 测试各方案在节点数不同的情况下加密过程的存储开销, 结果如图 6 所示。由于 2-Hop 标签结构紧凑, 本方案存储空间开销仅次于 SecGDB, 优于 PGAS、文献[27]的方案和 GES。

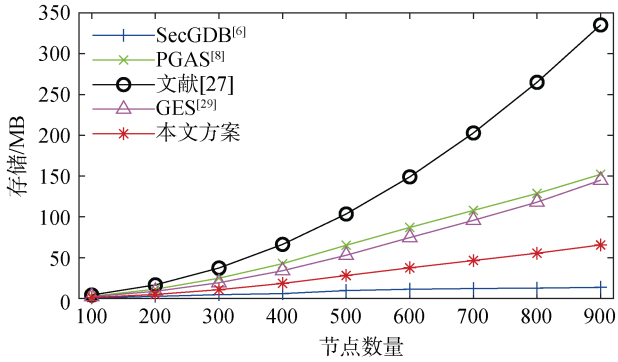



**图 6 加密图数据的存储开销**

**Figure 6 Storage cost of encrypted graph**

### 7.2.2 时间开销分析

本节对各方案在加密过程和查询过程的时间开销进行实验评估。

在加密过程中, 选取不同节点数构造符合要求的明文索引作为输入进行测试。各方案在选取不同节点数额的子图的情况下对索引进行加密所需时间如图 7 所示。

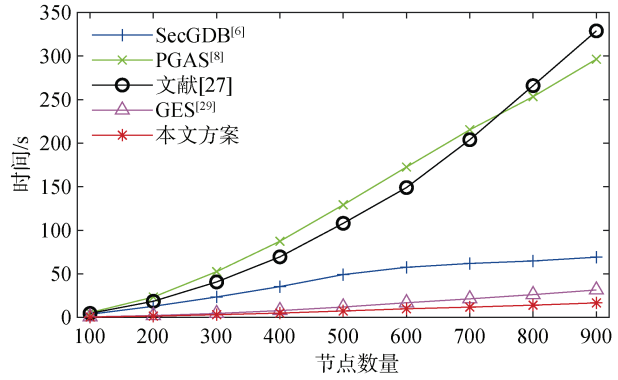



**图 7 图数据加密的时间开销**

**Figure 7 Time overhead of encrypting graph**

本文方案加密的时间开销主要取决于图的 2-Hop 标签索引大小和加密每一项所需时间。由于 2-Hop 标签结构紧凑, 且不同于 SecGDB、PGAS 和文献[27], 本文方案不使用公钥加密, 有最优的加密时间开销。

最短路径查询过程包括三部分: 生成令牌, 路径查询和解密。实验中, 本文方案令牌生成需计算 4 个伪随机函数, SecGDB[6]计算 3 个伪随机函数,

GES[29]计算 1 个伪随机函数, 经测试所需时间均小于 0.1ms, 但 PGAS[8]的令牌中包含一个基于 Paillier 加密的门限值, 生成令牌的时间开销约为 1.71ms, 而文献[27]的方案需要执行一次基于 Paillier 的 PIR 协议。

然后, 我们对 SecGDB[6]、PGAS[8]、文献[27]、GES[29]和本文方案的查询算法过程选取超过 1000 组起点 $s$ 和终点 $t$ 进行测试, 评估在不同节点数的网络上完成查询平均所需时间, 如图 8 所示。

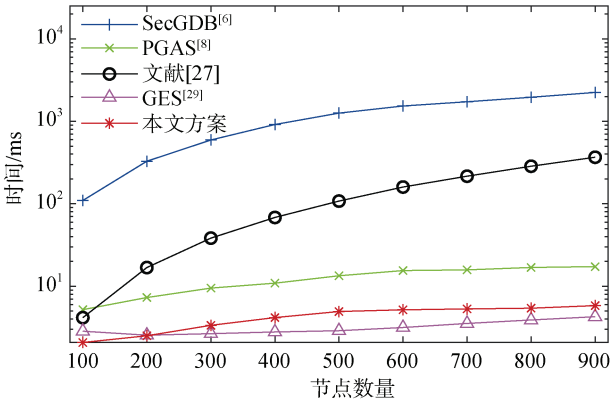



**图 8 查询时间开销**

**Figure 8 Time overhead of query process**

SecGDB 存储加密的邻接表结构, 存储开销很小, 但即使在历史查询数据结构辅助下, 在大量查询过后效率有极大提升, 查询过程仍然非常耗时。PGAS 是一个约束距离查询方案, 虽然不受路径长度影响, 但由于使用公钥密码算法进行计算和比较, 其时间开销不可避免地增加。文献[27]的方案需要执行 $l$ 次基于 Paillier 的 PIR 协议完成查询。GES 拥有最优的查询时间效率, 与 GES 相比, 我们的方案受到路径上节点的标签大小影响, 查询的时间开销有所增加, 但实际增加的时间开销在可接受范围内。

### 7.2.3 通信开销分析

本节对本文方案的通信开销进行分析, 并将其与 PGAS[8], 文献[27]和 GES[29]的通信开销进行比较。

数据拥有者完成对图数据的加密后, 将加密的图数据发送给云服务器, 这一过程是一次性的操作, 通信开销与加密的图数据的存储开销一致。

在查询过程中, 用户和云服务器之间的通信开销包括用户将令牌发送给云和云服务器返回加密结果给用户。本文方案的令牌包括 4 个伪随机函数值, 通信开销为 256B; SecGDB 令牌包括 3 个伪随机函数值, 通信开销约为 141B; PGAS 的令牌中包含 4 个哈希值和 1 个 PCTD 加密的门限值, 经测试平均通信开销为 1.46KB; GES 方案令牌仅为 1 个伪随机函数值, 通信开销为 64B。在云服务器将加密结果返回给用户的过程中, 通信开销的大小与返回的加密结果有关。PGAS 方案返回一个 PCTD 加密的距离值, 开销为距离值密文大小, 由于使用非对称加密, 通信开销约为 1.2KB。而 GES 和本文方案均返回 SKE 加密的最短路径, 在实验数据集上测试平均通信开销小于 200B。在 Ramezanian 等人[27]的方案中, 查询过程最多需执行 $l$ 次私有信息检索(PIR)协议, 云每次需要发送约 1KB 的 PIR 结果。所以这个阶段的通信开销是 $l$ KB。

通过上述分析可知, 本文提出的结构化加密图的最短路径查询方案具有较好的存储空间复杂度。在实际的加密和检索过程中, 利用 OPE 而不是同态加密进行距离值的计算和比较, 减少了存储和通信开销, 提高了查询效率。

## 8 讨论

本节对有待进一步研究的方向和相应的解决思路进行讨论。

**扩展到无向图** 本文方案的适用范围很大程度上依赖于生成的索引结构。因此, 虽然方案主要针对有向图设计, 但是我们只需在 2-Hop 标签索引的计算过程中将无向图的每一条无向边视作一对方向相反的有向边, 将无向图视作一个特殊的“有向图”, 对其运行 PLL 算法, 而后续的加密、查询等算法过程则无需进行改动, 能使方案的适用范围自然扩展到无向图。

**带权图** 剪枝的 BFS 主要适用于计算无权图的 2-Hop 标签。若要使方案适用于带权图, 唯一需要改变的是在 2-Hop 标签计算过程中将对节点执行的剪枝的 BFS 算法改为剪枝的 Dijkstra 算法, 其余部分如加密、查询等是通用的, 不需做改动。

**全最短路径查询** 全最短路径查询返回两节点间的所有最短路径。对于任意可达的节点 $s$ 和 $t$, 本文方案仅返回从 $s$ 到 $t$ 的一条最短路径。在未来的工作中, 我们可以考虑通过解决以下两个方面的问题实现全最短路径查询。其一, 本文基于 2-Hop 标签构造支持最短路径查询的索引, 该索引仅能确保至少能查找出一条最短路径, 因为其中一部分距离相同的标签项在 PrunedBFS 算法过程中被“剪枝”。因此, 可以对 PrunedBFS 算法进行修改, 对于满足 distQuery($v_k$, $u$, $L^{k-1}$)=$d[u]$的标签项不执行剪枝, 生成符合条件的标签索引。其二, 利用改进的保序编码算法 OPE 对距离值进行编码和比较, 该算法不能判断相等关系, 倘若有若干相等的最小值, 仅能确定找

出的是其中之一, 可以通过修改距离值的加密和比较方法, 实现安全的相等测试。

**动态更新** 本文基于 2-Hop 标签设计索引结构以实现加密图上的最短路径查询, 若要实现数据更新, 每次更新需对节点进行遍历再重新生成索引, 工作量较大, 更新的效率可能不理想。此前, 已有研究者对 2-Hop 标签的增量更新和减量更新[16,31]进行研究。在未来的工作中, 我们可以考虑从 2-Hop 标签的增量和减量更新维护技术[16,31]和动态结构化加密技术[18-19,32]实现方案的动态更新。

## 9 结论

本文提出一个结构化加密图上的高效的最短路径查询方案, 保证了较好的存储开销和查询效率。对于需要较低延迟的应用, 如社交网络图上的敏感搜索, 本文方案有更好的性能。通过将无向图的无向边视为两条方向相反的有向边, 运行 PLL 算法计算 2-Hop 标签, 本文方案可以将适用范围自然地扩展到无向图。

与可达性查询和距离查询相比, 最短路径查询是一个更复杂的过程。因为实现路径查询, 设计了基于 2-Hop 标签的索引结构, 也可以实现数据更新, 因为更新涉及对节点进行遍历再重新生成索引, 其效率会受到影响。在未来的工作中, 我们将进一步考虑从 2-Hop 标签的增量和减量更新技术[16,31]和动态结构化加密技术 [18-19,32]实现本文方案的动态更新。此外, 我们可以关注图的结构和内容特性, 对加密的图数据进行统计分析, 计算和分析图的中心度和聚集情况等, 进一步拓宽结构化加密图的功能, 改善其效率。

## 参考文献


[1] Low Y, Bickson D, Gonzalez J, et al. Distributed GraphLab[J]. *Proceedings of the VLDB Endowment*, 2012, 5(8): 716-727.

[2] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A System for Large-Scale Graph Processing[C]. *The 2010 ACM SIGMOD International Conference on Management of data*, 2010: 135-146.

[3] Han W S, Lee S, Park K, et al. TurboGraph: A Fast Parallel Graph Engine Handling Billion-Scale Graphs in a Single PC[C]. *The 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013: 77-85.

[4] Chase M, Kamara S. Structured encryption and controlled disclosure[C]. *The 16th International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2010*, 2010: 577-594.

[5] Meng X R, Kamara S, Nissim K, et al. GRECS: Graph Encryption for Approximate Shortest Distance Queries[C]. *The 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015: 504-517.

[6] Wang Q, Ren K, Du M, et al. SecGDB: Graph encryption for exact shortest distance queries with efficient updates[C]. *Financial Cryptography and Data Security - FC 2017*, 2017: 79-97.

[7] Shen M, Ma B L, Zhu L H, et al. Cloud-Based Approximate Constrained Shortest Distance Queries over Encrypted Graphs with Privacy Protection[J]. *IEEE Transactions on Information Forensics and Security*, 2018, 13(4): 940-953.

[8] Zhang C, Zhu L H, Xu C, et al. PGAS: Privacy-Preserving Graph Encryption for Accurate Constrained Shortest Distance Queries[J]. *Information Sciences*: *An International Journal*, 2020, 506(C): 325-345.

[9] Liu C, Zhu L H, Chen J J. Graph Encryption for Top-K Nearest Keyword Search Queries on Cloud[J]. *IEEE Transactions on Sustainable Computing*, 2017, 2(4): 371-381.

[10] Dijkstra E W. A Note on Two Problems in Connexion with Graphs[J]. *Numerische Mathematik*, 1959, 1(1): 269-271.

[11] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms, 3rd Edition[M]. MIT Press, 2009: 643-699.

[12] Cohen E, Halperin E, Kaplan H, et al. Reachability and Distance Queries via 2-Hop Labels[J]. *SIAM Journal on Computing*, 2003, 32(5): 1338-1355.

[13] Akiba T, Iwata Y, Yoshida Y. Fast Exact Shortest-Path Distance Queries on Large Networks by Pruned Landmark Labeling[C]. *The 2013 ACM SIGMOD International Conference on Management of Data*, 2013: 349-360.

[14] Wang Y, Wang Q, Koehler H, et al. Query-by-Sketch: Scaling Shortest Path Graph Queries on very Large Networks[C]. *The 2021 International Conference on Management of Data*, 2021: 1946-1958.

[15] Liu Z Y, Li L, Zhang M X, et al. Efficient Constrained Shortest Path Query Answering with Forest Hop Labeling[C]. *2021 IEEE 37th International Conference on Data Engineering*, 2021: 1763-1774.

[16] Farhan M, Wang Q, Lin Y, et al. Fast Fully Dynamic Labelling for Distance Queries[J]. *The VLDB Journal*, 2022, 31(3): 483-506.

[17] Song D X, Wagner D, Perrig A. Practical Techniques for Searches on Encrypted Data[C]. *Proceeding 2000 IEEE Symposium on Security and Privacy*. *S&P*, 2002: 44-55.

[18] Kamara S, Papamanthou C, Roeder T. Dynamic Searchable Symmetric Encryption[C]. *The 2012 ACM conference on Computer and communications security*, 2012: 965-976.

[19] Stefanov E, Papamanthou C, Shi E. Practical Dynamic Searchable Encryption with Small Leakage[C]. *Proceedings 2014 Network and Distributed System Security Symposium*, 2014: 1-15.

[20] Cao N, Yang Z Y, Wang C, et al. Privacy-Preserving Query over Encrypted Graph-Structured Data in Cloud Computing[C]. *2011 31st International Conference on Distributed Computing Systems*, 2011: 393-402.

[21] Yin S, Fan Z, Yi P, et al. Privacy-preserving reachability query services[C]. *Database Systems for Advanced Applications - DASFAA 2014*, 2014: 203-219.

[22] Liu C, Zhu L H, He X J, et al. Enabling Privacy-Preserving Shortest Distance Queries on Encrypted Graph Data[J]. *IEEE Transac-*

*tions on Dependable and Secure Computing*, 2021, 18(1): 192-204.

[23] Sun F Y, Yu J, Ge X R, et al. Constrained Top-k Nearest Fuzzy Keyword Queries on Encrypted Graph in Road Network[J]. *Computers & Security*, 2021, 111: 102456.

[24] Guan Y G, Lu R X, Zheng Y D, et al. Achieving Efficient and Privacy-Preserving ($\alpha$, $\beta$)-Core Query over Bipartite Graphs in Cloud[J]. *IEEE Transactions on Dependable and Secure Computing*, 2023, 20(3): 1979-1993.

[25] Aly A, Cuvelier E, Mawet S, et al. Securely solving simple combinatorial graph problems[C]. *Financial Cryptography and Data Security - FC 2013*, 2013: 239-257.

[26] Samanthula B K, Rao F Y, Bertino E, et al. Privacy-Preserving Protocols for Shortest Path Discovery over Outsourced Encrypted Graph Data[C]. *2015 IEEE International Conference on Information Reuse and Integration*, 2015: 427-434.

[27] Ramezanian S, Meskanen T, Niemi V. Privacy Preserving Shortest Path Queries on Directed Graph[C]. *2018 22nd Conference of Open Innovations Association*, 2018: 217-223.

[28] Bramm G, Schütte J. Cipherpath: Efficient traversals over homomorphically encrypted paths[C]. *The 17th International Joint Conference on e-Business and Telecommunications*, 2020: 271-278.

[29] Ghosh E, Kamara S, Tamassia R. Efficient Graph Encryption Scheme for Shortest Path Queries[C]. *The 2021 ACM Asia Conference on Computer and Communications Security*, 2021: 516-525.

[30] Curtmola R, Garay J, Kamara S, et al. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions[J]. *Journal of Computer Security*, 2011, 19(5): 895-934.

[31] Akiba T, Iwata Y, Yoshida Y. Dynamic and Historical Shortest-Path Distance Queries on Large Evolving Networks by Pruned Landmark Labeling[C]. *The 23rd international conference on World wide web*, 2014: 237-248.

[32] Lai R W F, Chow S S M. Parallel and dynamic structured encryption[C]. *12th International Conference of Security and Privacy in Communication Networks, SecureComm 2016*, 2016: 219-238.

**潘瑛颖** 于 2018 年在南京邮电大学数字媒体技术专业获得工学学士学位。现在福建师范大学网络空间安全专业攻读硕士学位, CCF 学生会员。研究领域为云存储安全、结构化加密、数据隐私保护。Email: whxxpyy@163.com

**陈兰香** 于 2009 年在华中科技大学计算机系统结构专业获得博士学位。现任福建师范大学教授、博士生导师, CCF 会员。主要研究领域为密码学及其应用、结构化加密、数据隐私保护、云计算与云存储安全。Email: lxiangchen@fjnu.edu.cn