

结构化加密的隐私集合并集协议

姜 鹏¹, 陈兰香^{1,2}

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

² 浙江理工大学 信息科学与工程学院 杭州 中国 310018

摘要 现实生活中,代理为具有特定属性的人群提供服务,云存储的发展使得代理能够从云端获取丰富的数据以提高服务效率。在社交网络图中,具有特定属性的人群的邻居节点集很可能具有相同的属性,为代理获取数据提供了潜在机会。随着社交网络图规模的持续增长,本地维护这些数据已变得越来越困难。云计算的快速发展为解决这一问题提供了有效的途径:通过将社交网络图数据上传至云服务器,可以显著节省本地资源,但与此同时也带来了数据安全问题。结构化加密(Structured Encryption, STE)技术为这一问题提供了一种可行的解决方案。STE 能够对社交网络图进行加密,并通过构建关联性的加密索引,支持对加密图的复杂查询操作。然而,结构化加密在查询效率和实用性上仍然存在一定的局限性。隐私集合并集(Private Set Union, PSU)协议能够在保护两方交集不被泄露的前提下求出两方集合的并集,使得代理从云端获取数据的过程中返回必要的数据并确保数据的安全性,我们提出了基于结构化加密的隐私集合并集协议(Private Set Union for Structured Encryption, STE-PSU),实现了服务器与代理两方的隐私集合并集操作,降低了查询过程中的计算和通信开销。方案引入一种不经意键值存储(Oblivious Key-Value Stores, OKVS)构造,即基于三个哈希函数的混淆布谷鸟哈希表(3-Hash Garbled Cuckoo Table, 3H-GCT),在保证隐私的基础上实现了两方高效的 PSU 协议。同时云服务器上的社交网络图通过结构化加密技术加密后存储,保证了数据的安全性。理论分析与实验结果表明,尽管我们提出的代理与服务器两方的 PSU 协议需要首先搜索加密索引再进行并集操作,但仍然比已有的 PSU 协议效率更优。安全性分析证明,本文方案满足自适应选择查询攻击(Adaptive Chosen Query Attacks, CQA2)安全性。

关键词 结构化加密; 云存储; 隐私集合并集协议; 不经意键值存储

中图法分类号 TP309.7 DOI 号 10.19363/J.cnki.cn10-1380/tn.2026.05.05

Private Set Union Protocol for Structured Encryption

JIANG Peng¹, CHEN Lanxiang^{1,2}

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

² School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China

Abstract In practical scenarios, proxies provide services to individuals with specific requirements. The development of cloud storage enables proxies to access abundant data from the cloud, thereby enhancing service efficiency. In a social network, neighbors of individuals with specific attributes often have similar attributes, presenting potential opportunities for proxies to acquire data. With the continuous growth of social network graph scales, maintaining these data locally has become increasingly challenging. The rapid development of cloud computing provides an effective solution to this issue by uploading social network graph data to cloud servers can significantly save local resources. However, this approach also introduces data security concerns. Structured Encryption offers a viable solution to this problem. STE enables the encryption of social network graphs and supports complex query operations on encrypted graphs by constructing associative encrypted indexes. Nevertheless, structured encryption still has certain limitations in terms of query efficiency and practicality. The Private Set Union (PSU) protocol enables the computation of the union of two sets while ensuring that the intersection of the two sets remains confidential. To allow the proxy to retrieve necessary data from the cloud while ensuring data security, we propose the Structured Encryption based Private Set Union Protocol (STE-PSU). This protocol achieves private set union operations between the server and the proxy, reducing both computational and communication overhead during the query process. The scheme introduces an Oblivious Key-Value Stores (OKVS) construction, i.e., a 3-Hash Garbled Cuckoo Table (3H-GCT), which achieves efficient PSU operation while preserving privacy. Additionally, the social network graph on the cloud server is stored securely through structured encryption. Theoretical analysis and experimental results show that, although the proposed PSU protocol requires an initial search of the encrypted index before performing the union operation, it is still more efficient than existing two-party PSU protocols. Security analysis verifies that the proposed scheme satisfies Adaptive Chosen Query Attacks (CQA2) security.

Key words structured encryption; cloud storage; Private Set Union Protocol(PSUP); Oblivious Key-Value Stores(OKVS)

通讯作者: 陈兰香, 博士, 教授, 博士生导师, 中国计算机学会(CCF)会员, Email: lxiangchen@gmail.com。

本课题得到国家自然科学基金(No. 62072105)、福建省自然科学基金(No. 2024J01068)项目资助。

收稿日期: 2024-12-02; 修改日期: 2025-02-14; 定稿日期: 2026-04-07

1 引言

大数据时代的到来使得本地存储资源变得越来越紧张, 云计算的快速发展有效地解决了这一问题, 将数据存储云端不仅便于管理数据, 而且有助于数据的共享^[1]。但与此同时云计算引发了数据的安全问题, 为了防止数据泄露, 需要将数据加密后上传至云端, 但这又降低了数据的可用性。为了解决这一挑战, Song 等人^[2]提出了第一个可搜索加密方案, 实现了对加密数据的搜索功能。此后, 可搜索加密在功能、效率 and 安全性方面都取得了重大进展^[3]。但这类方案只适用于文本数据, 无法实现对复杂数据加密后的检索。2010 年, Kamara 等人^[4]提出了第一个结构化加密(Structured Encryption, STE)方案, 实现了复杂数据类型的加密存储, 加密后仍保留对复杂数据的搜索能力。而社交网络图能够呈现复杂的社会关系, 并且其规模逐渐增大, 已难以在本地进行维护。由于云计算会带来数据安全问题, 我们能够通过结构化加密对社交网络图进行加密后外包给云服务器, 并实现对加密社交网络图的检索功能。

现实生活中, 代理为社交网络中具有特定属性的人群提供服务, 为提高实用性, 代理需要向云端请求以丰富本地数据集。例如, 在现实生活中, 房产中介、银行贷款机构等各类代理, 需要向云服务器发起查询, 以获取有购买或贷款需求的个人信息。我们的关键思想在于, 在社交网络图中, 具有某种属性节点的邻居节点很可能具有相同或相似的属性^[5], 因此代理希望向云端请求获得与某种属性节点相邻的节点数据。通常, 结构化加密方案^[4]能够支持基于可链接性的查询以获取相应的节点数据。然而, 若将所有查询结果不加筛选地返回, 会导致大量无关节点数据的传输, 这不仅增加了数据处理的计算负担, 还带来了显著的通信开销, 在实际应用场景中表现出较低的效率。为解决这一问题, 隐私集合求并(Private Set Union, PSU)原语提供了一种有效的解决方案。PSU 能够在仅计算发送方和接收方集合并集的同时, 确保不泄露任何额外的敏感信息。在本方案中, 我们将云服务器和代理分别配置为 PSU 协议的发送方和接收方: 其中, 代理维护社交网络中的节点数据集, 而服务器则提供基于特定属性查询所得的邻居节点集。通过将这两个集合作为输入执行隐私集合求并操作, 最终由代理获得并集结果, 从而实现了高效且隐私保护的数据查询与处理。令 m_p 表示代理端数据集, m_s 表示服务器端查询得到的数据集, 最终代理得到的结果为 $m_s \cup m_p - m_p$ 。因此 PSU 能

够实现服务器只将必要的节点密文数据返回给代理, 降低了计算和通信开销, 并且保护了两方的数据安全。为了提高做并集的效率, 本文通过一种高效的 OKVS 结构(3H-GCT^[6])来实现代理与服务器两方的隐私集合求并操作, 保证了效率 and 安全性。

具体而言, 本文提出了一种基于结构化加密的 PSU 协议, 定义为 STE-PSU 协议。本文的贡献概括如下:

(1) 我们将 PSU 协议应用于结构化加密中, 实现了代理与服务器两方的隐私集合求并操作。代理通过向数据拥有者请求授权, 并向服务器查询感兴趣关键词节点的邻居节点集, 与本地数据进行隐私集合求并操作, 返回必要的节点数据, 从而显著降低了计算和通信开销。

(2) 通过结构化加密后存储社交网络图, 并通过引入 3H-GCT 算法实现了并集操作, 除了代理得到的本地数据与搜索得到的邻居节点集的并集, 代理与服务器之间并没有向对方泄露其他任何信息, 确保了数据的安全性。

(3) 经过安全性分析证明, 本文方案满足 CQA2 安全性; 并通过在真实数据集上的实验表明, 本文代理与服务器两方的 PSU 协议具有很高的效率。

本文组织如下: 第二部分介绍云存储、可搜索加密、结构化加密和隐私保护集合求并的相关工作; 第三部分介绍本文的符号定义、系统模型、安全模型、不经意键值存储的概念、3H-GCT 算法和索引构建; 第四部分介绍本文的 PSU 协议设计; 第五部分介绍 STE-PSU 协议的详细算法并进行安全性分析; 第六部分对方案进行性能分析并对实验结果进行总结分析。第七部分对全文进行总结。

2 相关工作

信息技术的发展使得大量的数据需要存储和处理, 已经无法靠独立的计算机完成, 云计算的发展使得用户能够通过租赁存储空间, 便捷地管理和共享数据。但为了保护隐私, 用户需要将数据的密文上传至云服务器, 但这毫无疑问阻碍了数据的使用和共享, 如何在使用云存储的同时实现高效查询和共享数据是一个重要的问题。

可搜索加密。2000 年, Song 等人^[2]首次提出可搜索加密的概念, 该方案对文档中的关键字计算其特殊格式的哈希值, 并将其插入密文对应位置中用以执行顺序扫描。在如今的大数据时代, 当文档基数很大时, 这种检索方法效率极低。但他们的工作对可搜索加密有重要意义, 可搜索加密之后得到了广泛的

研究。2006 年, Curtmola 等人^[7]第一次提出了倒排索引的概念, 并给出两种 SSE 构造, 分别为针对非适应性敌手的 SSE-1 构造和针对适应性敌手的 SSE-2 构造。2017 年, Kim 等人^[8]提出了一种前向安全的动态可搜索对称加密方案, 索引采用倒排索引和前向索引的双字典数据结构, 能够支持删除数据, 显著提高了检索效率。Cash 等人^[9]提出了第一个支持布尔查询的可搜索加密方案, 称为 OXT 方案。2019 年, Du 等人^[10]基于 Cash 等人^[9]的工作, 提出了一种支持布尔查询的动态多用户可搜索加密方案, 数据所有者能够授权多个用户进行查询并限制其查询权限, 并能够高效地动态更新用户的查询权限。2018 年, Liu 等人^[11]提出了满足 UC 安全性的多用户可验证可搜索加密方案, 支持多个用户进行检索, 并能够动态地更新云上文档。2021 年, Zhang 等人^[12]设计了一种基于树结构的索引, 能够提高检索密文的效率, 但是同时降低了检索的成功率。2022 年, Gan 等人^[13]提出了可验证可搜索对称加密方案, 能够实现次线性的搜索时间开销, 并能够确保搜索结果的正确性和可验证性。2022 年, Zuo 等人^[14]基于文献[15]中的动态可搜索加密方案, 提出了一种支持范围查询的前向和后向安全的动态可搜索加密方案, 使用一种基于文献[15]改进后的二叉树索引。2023 年, Zhao 等人^[16]提出了基于属性的多用户协同可搜索加密的思想, 并在多用户场景下设计了一种基于属性的协同可搜索加密方案, 实现了对加密数据的细粒度访问控制。He 等人^[17]提出了具有常量客户端存储成本的动态可搜索加密方案, 包括前向安全的 CLOSE-F 方案和前向、后向安全的 CLOSE-FB 方案。Song 等人^[18]提出可以保护访问模式和搜索模式的可搜索加密方案, 主要思想为利用重加密密码系统来洗牌多个云上的索引条目。为了保护访问模式, 将安全索引分配到多个服务器, 通过一种索引重分配协议允许用户更新云上的索引。Huang 等人^[19]针对目前 SSE 方案中非适应性文件注入攻击可以有效恢复一些重要信息的问题, 提出新的安全属性即前向隐私, 并构造了满足前向隐私的高效 SSE 方案 Cetus。

结构化加密。2010 年, Kamara 等人^[4]扩展了之前可搜索加密的工作, 首次提出了适用于复杂结构数据的数据结构化加密(Structured Encryption, STE)。结构化加密允许用户对复杂数据加密后存储在云端, 并保持可以高效查询数据的能力。方案思想为: 将数据视为(1)数据的结构 δ 和(2)数据项 m 的结合, 分别对其进行加密得到数据的结构的密文 γ 和数据项的密文 c 。被授权的查询者发出查询 q 并接收令牌 τ , 根据

令牌 τ 和加密的结构 γ 可恢复指向关于查询 q 的密文项的指针, 再经过解密即可得到明文数据项 m 。

2013 年, Kamara 等人^[20]设计了一种基于结构化加密的特定用途的混淆电路(special-purpose garbled circuits), 加密方式可以满足通过令牌的方式查询的要求并且特定用途的混淆电路能够高效地实现基于图的各种功能。2022 年, Xue 等人^[21]提出了第一个具有 CQA2 安全性的知识图谱加密方案, 能够支持在加密的多关系图和属性图上进行搜索。2017 年, Liu 等人^[22]提出重要的加密图查询类型即 top- k 最近邻关键词查询, 给定关键词和所查节点, 能够返回图中 k 个距离所查节点距离最近且包含所查关键词的节点。Wang 等人^[23]提出了加密图上的最短路径查询, 并支持图数据的更新。2020 年, Wang 等人^[24]又提出了一种基于加法同态加密和混淆电路的图加密方案, 支持加密图上的最短路径和最大流问题等查询, 但方案的索引构建时间开销较大。2022 年, Wang 等人^[25]提出了支持双约束的隐私保护最短距离查询方案, 通过向图中添加伪边可以防止拓扑信息向云服务器泄露。2023 年, Liang 等人^[26]提出了 SqueezePIR 技术, 使得在结构化私有图上检索数据时能够不向服务器泄露图的结构信息, 并能高效地完成检索。

隐私保护集合求并。在明文下计算两个集合的并集是一个简单的操作, 因为不需要考虑隐私问题, 但若想在计算并集的同时确保不泄露其他任何数据, 就变得十分困难。隐私保护集合求并(Private Set Union, PSU)正是为了解决这一问题而提出的。PSU 和隐私保护集合求交(Private Set Intersection, PSI)功能相似, 但实现技术上却有明显不同, 构造 PSI 的技术不能直接应用于 PSU, 后者的实现更具有挑战性。主要原因在于计算两方集合并集的过程中, 不能泄露交集元素, 这使得 PSU 协议无法达到与最优 PSI 协议同等的效率。近年来, 相比较于 PSU 协议, PSI 协议取得了更重大的进展^[27-29]。文献[30]也指出了 PSU 协议隐藏交集元素的必要性: 由于单个黑名单列表往往存在缺陷, 通过 PSU 协议计算多个黑名单列表的并集能够更有效地抵御恶意攻击。各方的黑名单列表都是通过各自的检测策略生成的, 如果 PSU 协议泄露了交集元素, 好奇或恶意的各方则更可能推测出其他方的检测策略。

2017 年, Davidson 等人^[31]提出了基于布隆过滤器(BF)和加法同态加密技术(AHE)的 PSU 协议, 在他们的协议中, 接收者持有集合 X , 使用 k 个哈希函数将集合 X 映射到布隆过滤器中, 并将布隆过滤器中每一项的值进行取反, 再使用 AHE 方案对布隆过滤

器进行加密并发送给发送者, 发送者将自己集合中的每一项 y 使用 k 个哈希函数检索加密的布隆过滤器中 k 个项并进行同态加法运算得到 c , 并将 (cy, c) 返回给接收者, 接收者解密每个密文对, 当解密 $c \neq 0$ 时, 解密 cy 即可得到发送方独有的元素, 从而得到并集结果。2019 年, Kolesnikov 等人^[32]提出了基于反向私有成员测试协议(RPMT)和不经意传输(OT)的 PSU 协议。RPMT 协议基于多项式, 用来测试发送方的每个元素 y 是否属于接收者的集合 X , 故需要执行 n 次, 这会产生较大的开销, 因此他们采用分桶技术将两方的集合都映射到 $O(n \log n)$ 个桶中(这里 n 为平衡数据集的大小), 再在每两个桶之间进行 RPMT 协议和 OT 协议, 最终接收者得到并集结果。

2022 年, Jia 等人^[33]将 Shuffle 技术应用到 PSU 协议中, 针对参与两方数据集大小不平衡的情况设计了两种不同的 PSU 协议, 分别为打乱接收者集合的 Π_{PSU}^R 协议和打乱发送者集合的 Π_{PSU}^S 协议。2023 年, Zhang 等人^[6]分析了 Kolesnikov 等人^[32]的方案, 认为其效率的瓶颈就在于其中基于多项式构造的反向私有成员测试协议, 他们通过 3H-GCT 构造提出了多查询反向私有成员测试(MQ-RPMT)协议, 并经过实验与现有较高效的 PSU 协议进行对比论证其效率得到了极大的提升。Bienstock 等人^[34]提出了一种基于随机带矩阵的 RB-OKVS, 能够进一步降低计算和存储开销。

3 预备知识

本节介绍本文的符号定义、函数定义、系统模型、安全模型、不经意键值存储、3H-GCT 算法和索引构建。

3.1 符号定义

本文使用的符号如表 1 所示, 下面定义文中使用的几个函数:

- 伪随机置换 $f: \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \{0,1\}^*$.
- 伪随机函数 $g: \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \{0,1\}^{\log(n)+\lambda}$.
- 生成密钥的伪随机函数 P :

$$\{0,1\}^\eta \times \{0,1\}^* \rightarrow \{0,1\}^\lambda.$$

定义 1(对称密钥加密). 对称密钥加密算法由加密算法 Enc 和解密算法 Dec 组成, Enc 和 Dec 使用相同的密钥 k_i , 表示为 $\Phi = \{\text{Enc}, \text{Dec}\}$ 。

$\text{Enc}_{k_i}(m) \rightarrow c$: 一种概率性算法, 输入密钥 k_i 对明文 m 进行加密得到密文 c 。

$\text{Dec}_{k_i}(c) \rightarrow m$: 一种确定性算法, 输入密钥 k_i

对密文 c 进行解密得到明文 m 。

表 1 符号定义

Table 1 Symbol definition

符号	说明
G	无向图
λ	安全参数
η	伪随机函数 P 的密钥长度
$K = \{k_1, k_2, k_3, k_4\}$	密钥集合
W	关键词的集合
$m = \{m_1, \dots, m_i\}$	节点数据集
$v = \{v_1, \dots, v_i\}$	半保密项数据集
c	密文集合
I^*	倒排索引
L^*	邻居节点表
$\max_{I^*}$	I^* 中节点最大值
$\max_{L^*}$	L^* 中邻居节点最大值
τ	查询令牌
$R(w)$	I^* 中查询 w 的结果集
L_w	关键词 w 节点的邻居节点集
$\mathcal{K} \times \mathcal{V}$	键值对集合
S	3H-GCT 算法得到的向量 S
$\mathcal{G}$	cuckoo 图
u_i	cuckoo 图中的顶点
a	cuckoo 图中节点数量
m_p	代理本地数据集
R	加密结果集

定义 2(诱导置换). 诱导置换(Induced Permutation)的作用是破坏节点数据项与密文项的相关性, 作用范围为 $[n] \rightarrow [n]$ 。设 π 是 $[n]$ 上的诱导置换, 对所有的节点数据项 m_i , 有 $j \leftarrow \pi[i]$, 假设使用对称密钥 k_4 对 m_i 进行加密, 则密文项 $c_j := \Phi.\text{Enc}_{k_4}(m_i)$, 且数据项 $m_i := \Phi.\text{Dec}_{k_4}(c_j)$, 则称 π 为数据项与密文项上的诱导置换。诱导置换的引入可以向敌手隐藏访问模式。

3.2 系统模型

传统的两方 PSU 协议主要旨在保护双方隐私的前提下计算两方集合的并集, 其协议设计仅涉及发送方和接收方两个实体。然而, 本文创新性地引入 OKVS 技术引入结构化加密的社交网络图查询中, 以显著提升结构化加密方案的效率与实用性。与传统的两方模型不同, 本文提出的系统模型包含三个核心实体: 代理(Proxy)、数据拥有者(Data Owner)和云服务器(Cloud Server), 如图 1 所示。这一扩展模型不仅继承了 PSU 协议的隐私保护特性, 还通过引入

OKVS 技术进一步优化了数据查询与处理的性能, 从而为实际应用提供了更高效的解决方案。数据拥有者将社交网络图的密文上传至云服务器, 代理通过向数据拥有者请求授权进而与服务器之间运行 PSU 协议。

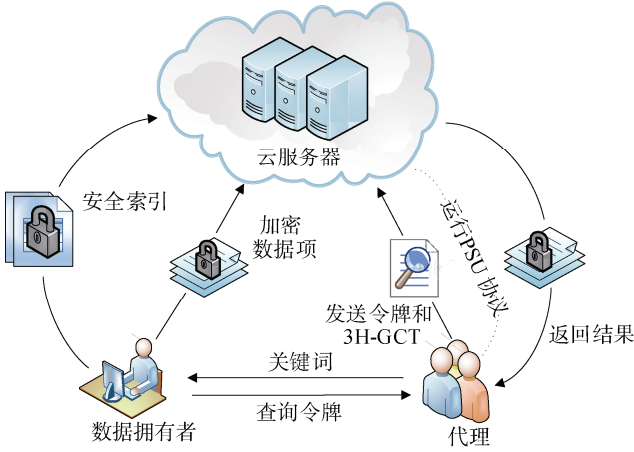


图1 系统模型

Figure 1 The system model

STE-PSU 协议包括五个步骤, 定义为五元组 $\Pi = (\text{GenKey}, \text{BuildIndex}, \text{Encrypt}, \text{Token}, \text{RunPSU})$, 其中, RunPSU 为代理与服务器之间运行 PSU 协议, 每个算法的功能描述如下。

(1) $\text{GenKey}(1^\lambda) \rightarrow K$: 数据拥有者输入安全参数 λ , 输出密钥集合 K , 用于加密索引表、3H-GCT 中的随机串和节点数据项。

(2) $\text{BuildIndex}(G, K, \mathbf{m}) \rightarrow (I^*, L^*)$: 数据拥有者输入密钥集合 K 和图 G , 构建图的倒排索引 I^* 和邻居节点表 L^* 。

(3) $\text{Encrypt}(k_4, \mathbf{m}^*) \rightarrow \mathbf{c}$: 数据拥有者使用密钥 k_4 对节点数据项 $\mathbf{m}^*$ 进行加密得到密文集合 $\mathbf{c}$ 。

(4) $\text{Token}(K, w) \rightarrow \tau$: 数据拥有者对代理的查询返回对应的加密令牌 τ , 用于与服务器之间运行 PSU 协议。

(5) $\text{RunPSU}(\mathbf{m}_p, \tau) \rightarrow \mathbf{m}_j \cup \mathbf{m}_p$: 运行代理与服务器之间的 PSU 协议。代理将由 $\mathbf{m}_p$ 生成的 3HGCT、令牌 τ 以及随机串 s 发送到服务器, 服务器经过搜索、解码、解密等操作得到结果集 R 并返回给代理, 代理使用密钥 k_4 解密即可得到并集结果。

3.3 安全模型

泄露函数. 我们给出两个泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 准确衡量由密文和查询过程中造成的泄露, 具体定义如下:

泄露函数 $\mathcal{L}_1$: 表示敌手可以从 $(G, I^*, L^*, \mathbf{m}, \mathbf{v})$ 学习到查询关键词 w 得到的密文项个数, 返回的结果集大小等信息, 即 $\mathcal{L}_1(G, I^*, L^*, \mathbf{m}, \mathbf{v}) = \{|R(w)|, |L_w|, |R|\}$ 。

泄露函数 $\mathcal{L}_2$: 由搜索模式和访问模式组成, 即 $\mathcal{L}_2(G, q) = \text{SP}(q) + \text{AP}(q)$ 。

- 搜索模式 $\text{SP}(q)$: 给定一系列查询 q , 搜索模式 $\text{SP}(q)$: 表示敌手可以检查当前查询是否与之前的查询相同。对于任意的查询 q , $\text{SP}(q)$: 表示一个长度为 t 的二进制向量, 若 $q = q_i$, 则向量第 i 个元素置为 1, 否则置为 0。

- 访问模式 $\text{AP}(q)$: 给定一系列查询 q , 访问模式 $\text{AP}(q)$ 显示了两次查询是否包含相同的密文项, 而本方案通过引入诱导置换破坏了数据项与密文项的对应关系。

本文方案采用自适应选择查询攻击 (Adaption Chosen Query Attacks, CQA2)^[4] 模型进行安全性分析, 通过泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 来证明本文方案满足 CQA2 安全性。在 CQA2 安全模型中, 敌手 $\mathcal{A}$ 适应性地发起查询。

定义 3(CQA2 安全性). 对于给定的 STE-PSU 方案 $\Pi = (\text{GenKey}, \text{BuildIndex}, \text{Encrypt}, \text{Token}, \text{RunPSU})$, 一个安全参数 λ , 一个敌手 $\mathcal{A}$ 和一个模拟器 $\mathcal{S}$, 其中泄露函数为 $\mathcal{L}_1$ 和 $\mathcal{L}_2$, 定义游戏 Real 和 Ideal 如下所示。

$\text{Real}_{\mathcal{A}}(\lambda)$: 敌手 $\mathcal{A}$ 首先生成图 G 的索引表 I^* 和 L^* 。挑战者运行 $K \leftarrow \text{GenKey}(1^\lambda)$ 生成密钥集合 K , 运行 $\mathbf{c} \leftarrow \text{Encrypt}(k_4, \mathbf{m}^*)$ 得到密文 $\mathbf{c}$ 。敌手 $\mathcal{A}$ 自适应地发起查询 q , 挑战者生成令牌 $\tau \leftarrow \text{Token}(K, w)$ 发送给 $\mathcal{A}$ 。最后敌手返回一个 bit 位 b 。

$\text{Ideal}_{\mathcal{A}, \mathcal{S}}(\lambda)$: 敌手 $\mathcal{A}$ 首先生成图 G 、节点数据项集合 $\mathbf{m}$ 和半保密数据项集合 $\mathbf{v}$ 。模拟器 $\mathcal{S}$ 根据泄露函数 $\mathcal{L}_1(G, I^*, L^*, \mathbf{m})$ 生成索引表 I^* 和 L^* 并发送给敌手 $\mathcal{A}$ 。敌手 $\mathcal{A}$ 自适应地发起查询。对每一个查询, 模拟器 $\mathcal{S}$ 根据挑战者提供的 $\mathcal{L}_2(G, q)$ 生成令牌 τ 并发送给敌手 $\mathcal{A}$ 。最后 $\mathcal{A}$ 返回一个 bit 位 b 。

如果对于所有多项式时间的敌手 $\mathcal{A}$, 总存在一个多项式时间的模拟器 $\mathcal{S}$ 满足公式:

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

那么该加密方案是自适应选择查询攻击 (CQA2) 安全的, 其中 $\text{negl}(\cdot)$ 是一个可忽略函数。

3.4 不经意键值存储

不经意键值存储(Oblivious Key-Value Stores, OKVS)的主要作用是存储键值对, 一个不经意键值存储由 Encode 算法和 Decode 算法组成。Encode 算法输入键值对集合 $\{(x_i, y_i)\}_{i \in [n]} \subseteq \mathcal{K} \times \mathcal{V}$, 返回键值对存储结构 S , Decode 算法输入 S 和键 x , 返回对应的值 y 。Decode 算法的输入可以是任意的 x^* , 当输入的 $x^* \in \mathcal{K}$ 时, 算法返回对应的值, 否则返回随机值。不经意键值存储最基本的性质为若 y 是随机值, 则 S 隐藏了 x 的值。首先给出键值存储(Key-Value Storage, KVS)的定义。

一个键值存储结构由键值对集合 $\mathcal{K} \times \mathcal{V}$ 和函数集合 H 组成。

$\text{Encode}_H(\{(x_1, y_1), \dots, (x_n, y_n)\})$: 输入键值对集合 $\{(x_i, y_i)\}_{i \in [n]} \subseteq \mathcal{K} \times \mathcal{V}$, 输出键值对存储结构 S (或者以可忽略的概率输出 $\perp$)。

$\text{Decode}_H(S, x)$: 输入键值对存储结构 S 和键 x , 输出值 $y \in \mathcal{V}$ 。

上述算法中, Encode 算法是否成功输出 S 取决于函数 H 和键 x , 与值 y 无关, 因为 y 本身是随机值。这一点不同于多项式, 多项式编码总是能够成功。

正确性. 若一个键值存储结构是正确的, 则对于所有的 $A \subseteq \mathcal{K} \times \mathcal{V}$, 用不同的键 x :

$$(x, y) \in A \text{ and}$$

$$\perp \neq S \leftarrow \text{Encode}_H(A) \Rightarrow \text{Decode}_H(S, x) \rightarrow y$$

不经意性. 若键值对存储结构对所有不同的键的集合 $\{x_1^0, \dots, x_n^0\}$ 和 $\{x_1^1, \dots, x_n^1\}$, Encode 算法输出不为 $\perp$, 且 $\{S | y_i \leftarrow \mathcal{V}, i \in [n], \text{Encode}_H((x_1^0, y_1^0), \dots, (x_n^0, y_n^0))\}$ 与 $\{S | y_i \leftarrow \mathcal{V}, i \in [n], \text{Encode}_H((x_1^1, y_1^1), \dots, (x_n^1, y_n^1))\}$ 的分布是不可区分的, 则该键值存储就可以称为不经意键值存储。通常不经意性表明当值是随机的时, 生成的键值对存储结构 S 与集合 K 是不相关的。

随机性. 不经意键值存储满足随机性。对于 $A = \{(x_1, y_1), \dots, (x_n, y_n)\}$ 和 $x^* \notin \{x_1, \dots, x_n\}$, 若 $S \leftarrow \text{Encode}_H(A)$, 则 $\text{Decode}_H(S, x^*)$ 在集合 $\mathcal{V}$ 上是均匀分布的。

3.5 3H-GCT

文献[35]概括了满足不经意键值存储概念的多构造, 包括多项式、混淆布隆过滤器和 PaXoS^[36] (Probe-and-XOR of strings)。尽管多项式被广泛应用于许多 PSI 协议中, 但其缺点在于, 当输入集合较大

时, 时间开销会显著增加。Dong 等人^[37]将混淆布隆过滤器用于 PSI 协议中, 然而其主要缺陷是长度过大, 会导致较大的通信开销。文献[36]基于两个哈希函数的布谷鸟哈希表, 提出了一种在恶意模型下高效的 PSI 协议。而文献[38]通过分析指出, 混淆布谷鸟哈希表若采用三个哈希, 能够显著提高性能。文献[35]提出了新的 OKVS 构造即基于三个哈希函数的布谷鸟哈希表(3H-GCT), 并通过实验数据表明新的 OKVS 构造能显著提高现有 PSI 协议的效率。文献[6]基于 3H-GCT 提出了多查询反向私有成员测试协议, 实现了 PSU 协议线性的计算和通信开销。下面我们介绍基于剥离方法的 3H-GCT 构造。

剥离(Peeling). 3H-GCT 结构基于剥离思想。得到的键值对存储结构是一种超图 $\mathcal{G}_{3, n, a}$, 包含 n 条边和 a 个顶点, 每条边与三个顶点相连。该结构使用三个哈希函数 h_1, h_2, h_3 , 将每个键 x 映射到超边 $(u_{h_1(x)}, u_{h_2(x)}, u_{h_3(x)})$ 。编码键值对 (x, y) 到超图 $\mathcal{G}_{3, n, a}$ 中, 能够满足 $y = \mathcal{G}[u_{h_1(x)}] \oplus \mathcal{G}[u_{h_2(x)}] \oplus \mathcal{G}[u_{h_3(x)}]$, 即将键 x 在超图中对应的三个顶点的值异或得到对应的值 y 。剥离思想描述如下: 假设键 x 对应超边 $(u_{h_1(x)}, u_{h_2(x)}, u_{h_3(x)})$, 且顶点 $u_{h_2(x)}$ 无其他超边与其相连, 那么可以为超图中其他顶点分配值, 包括顶点 $u_{h_1(x)}$ 和 $u_{h_3(x)}$, 最后为顶点 $u_{h_2(x)}$ 分配值以满足 $y = \mathcal{G}[u_{h_1(x)}] \oplus \mathcal{G}[u_{h_2(x)}] \oplus \mathcal{G}[u_{h_3(x)}]$ 。这表明我们可以从超图中移除这条超边, 将其入栈, 首先使得其他键值对映射正确, 之后再将其出栈, 为顶点 $u_{h_2(x)}$ 分配值使得键值对 (x, y) 映射正确。并且移除超边会降低其他顶点的度, 这将有助于移除超图中其他的超边。

我们举例描述剥离的过程, 如图 2 所示。假设存在键值对 $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$ 需要存储, 键经过哈希函数映射的情况如图所示, 首先选择度为 1 的顶点 $S[6]$, 将 x_1 对应的超边入栈, 此时 $S[2]$ 度为 1, 再将 x_2 对应的超边入栈, 随后依次将剩下的超边入栈。对顶点的分配值阶段与剥离过程相反, 首先从栈中弹出 x_4 , 为顶点 $S[1]$ 、 $S[4]$ 、 $S[5]$ 分配值使键值对 (x_4, y_4) 映射正确, 再从栈中弹出 x_3 , 对顶点 $S[3]$ 分配值使得键值对 (x_3, y_3) 映射正确, 随后依次弹出剩下的键, 并对相应顶点分配值使得键值对映射正确。

处理 2-core 子图. 如果剥离过程能够移除图中所有顶点, 则该图称为可剥离的。然而在剥离过程中可能出现所有顶点的度都至少为 2 的情况, 称此时

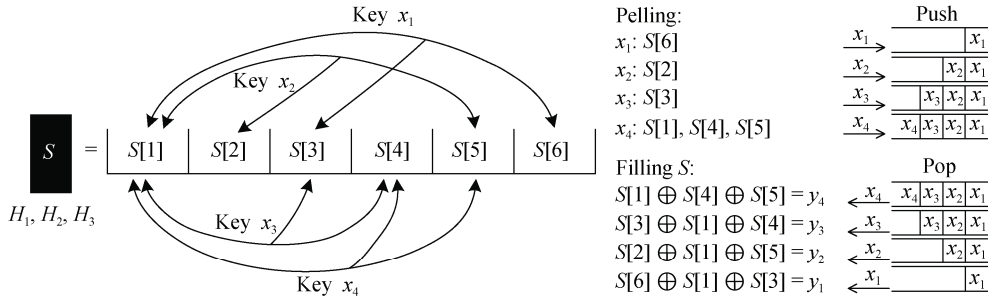


图 2 剥离的过程

Figure 2 The process of Peeling

的图为超图 $\mathcal{G}_{3,n,a}$ 的 2-core 子图。下面描述文献[35]中的处理方法。令 $\chi(n)$ 为 2-core 子图中的顶点数量, 并定义 $d(n)$ 为 $\chi(n)$ 的上界(实际上 $d(n)$ 非常小)。使用向量 $S = S_L \parallel S_R$, S_L 表示原本的超图, S_R 表示新增的 $d(n) + \lambda$ 个顶点。函数 $l_1(x)$ 有 3 个比特设为 1, 能够将键 x 映射到 S_L 中的三个顶点, 函数 $l_2(x)$ 是一个随机的字符串, 能够将键 x 映射到 S_R 中的若干顶点。最终键 x 对应的值由 S_L 中的三个顶点的值与 S_R 中若干顶点的值异或得到。因此编码过程必须为 S 分配正确的值来符合键值对存储的正确性。

3.6 索引构建

本文方案的安全索引由数据所有者构建, 数据所有者拥有丰富的社交网络图, 通过结构化加密构建关键词倒排索引 I^* 和邻居节点表 L^* , 以实现两者之间可链接的查询。对每一个关键词 w , 生成包含该关键词的节点数据项集合, 将其连同半保密数据项一起使用 f 和 g 进行加密得到倒排索引; 为每个节点的半保密数据项生成该节点的邻居节点表。具体构造过程见算法 5, 我们在图 3 中给出一个具体的示例, 图 3(a) 为一个具体的社交网络图, 图 3(b) 为倒排索引, 图 3(c) 为邻居节点表。给定关键词 w , 生成令牌

$\tau := (f_{k_1}(w), g_{k_2}(w))$, 使用 $f_{k_1}(w)$ 找到倒排索引入口, 并使用 $g_{k_2}(w)$ 与对应的值异或得到 $\langle \pi(i), v_i \rangle$ 。使用半保密数据项 v_i 作为子令牌查询索引表 L^* , 返回查询结果 L_w 。

4 PSU 协议设计

在双方 PSU 协议中, 发送者(Sender)与接收者(Receiver)各自持有私有集合, 通常由接收者得到并集结果。而在我们的模型中, 代理(Proxy)作为 PSU 协议的接收者, 云服务器(Cloud Server)作为 PSU 协议的发送者, 并集结果由代理获得, 丰富了代理端的数据集, 并且保护了两方的隐私。本文 PSU 协议基于 3H-GCT 算法实现, 下面分别介绍 3H-GCT 算法和本文的 PSU 协议设计。

4.1 3H-GCT 算法

STE-PSU 协议基于 3H-GCT 算法实现, 该算法输入键值对集合, 输出向量 S , 详细描述见算法 1。Parameters 算法初始化了一系列编码和解码算法的参数, 并定义三个哈希函数 h_1 、 h_2 和 h_3 , 以及 l_2 用于编码和解码算法。Encode 算法中, 第 9 行初始化长度为 a 的向量 S ; 第 11 行为剥离的过程, 将所有可剥

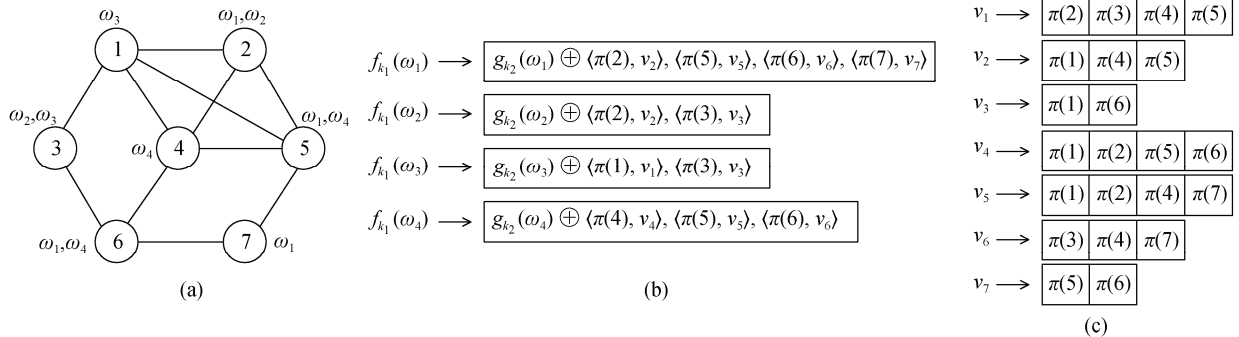


图 3 社交网络图及其索引表

Figure 3 Social network graph and its index table

离的超边依次入栈; 第 12 行处理 2-core 子图, 通过 l_1 和 l_2 定义向量中的值; 第 13 行 while 循环将栈中超边依次弹出, 并为对应顶点分配值使得键值对映射正确。Decode 算法输入向量 S 和键 x , 通过 l_1 和 l_2 将键 x 在向量 S 中对应位置的值进行异或得出结果并输出。

算法 1. 3H-GCT 算法

```

1: Parameters:
2: Security parameter  $\lambda$ , input length  $n$ .
3: A finite group  $\mathbb{G}$ .
4: Random functions  $h_1, h_2, h_3: \{0,1\}^* \rightarrow [m']$  and  $l_2(\cdot): \{0,1\}^* \rightarrow \{0,1\}^{d+\lambda}$ .
5: Parameters  $m' = 1.3n$  and  $d = 0.5\log n$ , where  $d$  upper bound the size of 2-core a  $(m', n)$ -Cuckoo graph.
6: Output length  $a = m' + d + \lambda$ .
7: Encode ( $\{(x_1, y_1), \dots, (x_n, y_n)\}$ ):
8: Define  $l_1(x) \in \{0,1\}^{m'}$  to be all zeroes except 1s at positions  $h_1(x), h_2(x), h_3(x)$ . Let  $\text{row}(x) := l_1(x) \parallel l_2(x)$ .
9: Initialize empty vectors  $S_L \in \mathbb{G}^{m'}$  and  $S_R \in \mathbb{G}^{d+\lambda}$ , let  $S = S_L \parallel S_R$ .
10: Initialize stack  $P$ .
11: While there is a node  $j \in [m']$  such that the set  $\{x_i \notin P \mid j \in h_1(x_i), h_2(x_i), h_3(x_i)\}$  is a singleton: Let  $x_i$  be the element of that singleton, and push  $x_i$  onto  $P$ .
12: Solve the system of equations  $S_L(l_1(x_i)) \oplus S_R(l_2(x_i)) = y_i$  for  $x_i \notin P$ , and assign the solutions to the corresponding locations in  $S$ .
13: WHILE  $P$  not empty DO
14:   (1) Pop  $x_i$  from  $P$ .
15:   (2)  $S_L$  is undefined in at least one of the positions  $h_1(x_i), h_2(x_i), h_3(x_i)$ . Set the undefined position(s) so that
      
$$S_L(l_1(x_i)) \oplus S_R(l_2(x_i)) = y_i.$$

16: END WHILE
17: Set any empty position in  $S$  with a random value from  $\mathbb{G}$ .
18: Output  $S$ .
19: Decode( $S, x$ ):
20: Return  $S_L(l_1(x)) \oplus S_R(l_2(x))$ .

```

4.2 PSU 协议

本文 PSU 协议进行代理与服务器间的隐私集合

求并操作, 代理端持有节点数据集 m_P , 而服务器端数据集通过以下方式得出: 代理将某一关键词的令牌 τ 发送到云服务器, 云服务器通过检索倒排索引和邻居节点表得到邻居节点集 L_w 。算法 RunPSU 包含三步算法 (GenOKVS, Query, Decrypt), 输入数据集 m_P 和令牌 τ , 输出结果集 R 。下面详细叙述每一步算法的过程。

4.2.1 GenOKVS (k_3, m_P) $\rightarrow S$

代理将本地数据集 m_P 使用 3H-GCT 结构进行存储, 详细描述如算法 2 所示。第 2 行 for 循环对 m_P 中的每一项 m_i 进行如下操作: 对节点标识符进行诱导置换 $j \leftarrow \pi[i]$, 并对随机串 s 使用密钥 k_3 加密得到 s_j , 组成键值对 (j, s_j) 存储在 D 中。第 7 行调用 3H-GCT 算法的 Parameters 算法进行初始化; 第 8 行调用 3H-GCT 算法的 Encode 算法, 输入 D 中的所有键值对, 输出向量 S 。

算法 2. GenOKVS (k_3, m_P) $\rightarrow S$

输入: 密钥 k_3 和节点集合 m_P

输出: 一个混淆布谷鸟哈希表 S

```

1: Initialize a random string  $s$  and a dictionary  $D$ .
2: FOR  $i = 1 : n$  DO
3:    $j \leftarrow \pi[i]$ ;
4:    $s_j \leftarrow \Phi.\text{Enc}_{k_3}(s)$ ;
5:   Store  $s_j$  into  $D[j]$ ;
6: END FOR
7: Parameters( $\cdot$ );
8:  $S \leftarrow \text{Encode}(j, s_j)$ , where  $(j, s_j)$  are all key-value pairs in  $D$ ;
9: Output  $S$ .

```

4.2.2 Query(τ, S) $\rightarrow R$

代理将向量 S 、令牌 τ 和随机串 s 发送到云服务器, 云服务器进行查询、解码、解密等操作, 详细描述如算法 3 所示。第 3 行使用 $f_{k_1}(w)$ 检索 I^* 得到标签项与 $g_{k_2}(w)$ 进行异或得到 $R(w)$; 第 5 行使用 $R(w)$ 中的半保密数据项检索 L^* 得到 L_w ; 第 6 行 for 循环为服务器调用 3H-GCT 算法的 Decode 算法, 输入向量 S 和 L_w 中的每一个元素, 并对输出结果使用密钥 k_3 进行解密, 若解密结果不等于 s , 则将对应的节点密文项添加到结果集 R 中, 最后将 R 返回给代理。

算法 3. $\text{Query}(\tau, S) \rightarrow R$ 输入: 令牌 τ 和混淆布谷鸟哈希表 S 输出: 集合 R

```

1:  Parse  $\tau$  as  $(f_{k_1}(w), g_{k_2}(w))$ ;
2:  Initialize a set  $R$ ;
3:   $R(w) \leftarrow I^*(f_{k_1}(w)) \oplus g_{k_2}(w)$ ;
4:  Parse  $R(w)$  as  $\langle J, v \rangle$ ;
5:   $L_w \leftarrow L^*(v)$ ;
6:  FOR  $j \in L_w$  DO
7:     $s_j^* \leftarrow \text{Decode}(S, j)$ ;
8:    Compute  $s'_j = \Phi.\text{Dec}_{k_3}(s_j^*)$ ;
9:    IF  $s'_j \neq s$  THEN
10:      Store  $c_j$  into  $R$ ;
11:    END IF
12:  END FOR
13:  Output  $R$ .
```

需要注意的是, L_w 中元素是通过检索 L^* 得到的, 而 L^* 的索引项均是节点经过诱导置换加密后的值, 因此服务器对 L_w 中的元素进行解码和解密的过程不会向服务器暴露任何有用的信息。

4.2.3 Decrypt(k_4, R) $\rightarrow m_j \cup m_p$

算法的执行为代理将结果集 R 中的每一项密文

使用密钥 k_4 进行解密, 即 $m_j = \Phi.\text{Dec}_{k_4}(c_j)$, 并将 m_j 与 m_p 合并得到最终的并集结果。

算法 RunPSU 包含 (GenOKVS, Query, Decrypt), 如图 4 (a)所示。首先代理端执行 GenOKVS 算法, 将令牌、3H-GCT 和随机串 s 发送到云服务器, 云服务器执行 Query 算法得到结果集 R 并返回给代理, 最后代理执行 Decrypt 算法即可得到并集结果。

5 STE-PSU 详细算法**5.1 算法描述**

本文 STE-PSU 方案主要包含五步算法: (GenKey, BuildIndex, Encrypt, Token, RunPSU), 其中算法 (GenKey, BuildIndex, Encrypt) 由数据拥有者执行, Token 算法的执行为代理向数据拥有者请求授权, 之后运行 RunPSU 算法进行代理与服务器之间的隐私集合求并操作。STE-PSU 算法过程如图 4 (b)所示。

5.1.1 GenKey(1^λ) $\rightarrow K$

数据拥有者输入安全参数 λ 和随机密钥 k , 伪随机函数 P 生成密钥集合 $K = (k_1, k_2, k_3, k_4)$, 详细描述见算法 4。 k_1, k_2 分别为加密索引表的伪随机置换 f 和伪随机函数 g 的密钥, k_3, k_4 分别为加密 3H-GCT 中的随机串和节点数据项的密钥。

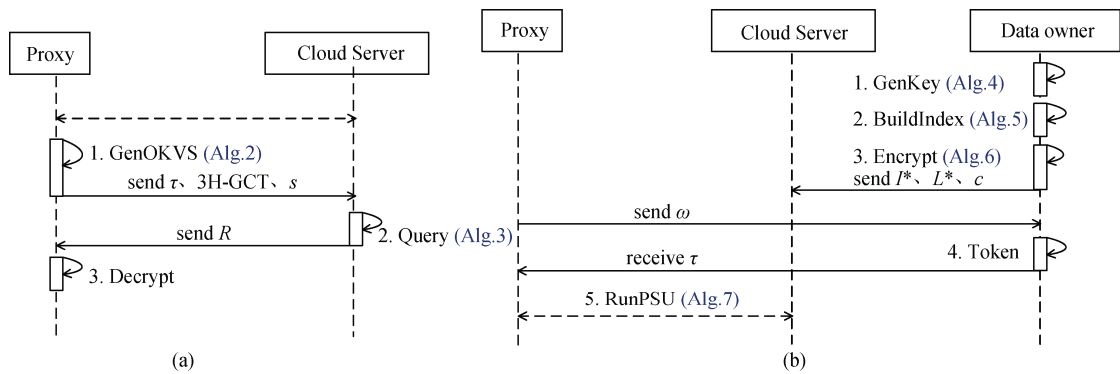


图 4 PSU 协议设计和 STE-PSU 算法

Figure 4 PSU protocol design and STE-PSU algorithm

算法 4. $\text{GenKey}(1^\lambda) \rightarrow K$ 输入: 安全参数 λ 输出: 密钥集合 K

```

1:  FOR  $i = 1 : n$  DO
2:     $k_i \leftarrow P_k(\lambda)$ ;
3:  END FOR
```

```

4:  Output  $K = (k_1, k_2, k_3, k_4)$ 
```

5.1.2 BuildIndex(G, K, m) $\rightarrow (I^*, L^*)$

数据拥有者生成图 G 的倒排索引和邻居节点表, 详细描述见算法 5。第 3 行对节点数据项 m 进行诱导置换得到 m^* ; 第 7 行 for 循环对关键词集合 W 中的每个关键词 w_i , 生成包含 w_i 的节点标识符集合存

储到 $I(w_i)$ 中; 并对集合进行填充至最大长度 $\max_{L^*}$, 使用伪随机函数 g 进行加密并存储到 $I^*(f_{k_1}(w_i))$ 中; 第 14 行 for 循环生成每个节点的邻居节点列表并填充至最大长度 $\max_{L^*}$; 最后输出 I^* 和 L^* 。

算法 5. BuildIndex(G, K, m) $\rightarrow (I^*, L^*)$

输入: 图 G 、数据项 m 和半保密数据 v

输出: 加密索引 I^* 和 L^*

```

1: Initialize dictionaries  $I$  and  $L$ ;
2: Parse  $G$  as  $(V, E)$ ;
3: FOR  $m_i \in m$  DO
4:    $j \leftarrow \pi(i)$ ;
5:    $m_j^* \leftarrow m_i$ 
6: END FOR
7: FOR  $w_i \in W$  DO
8:   FOR  $i = 1:|m|$  DO
9:     IF  $w_i \in m_i$  THEN
10:      Store  $\langle \pi(i), v_i \rangle$  into  $I(w_i)$ ;
11:    END IF
12:  END FOR
13: END FOR
14: FOR  $i = 1:|m|$  DO
15:   Generate the set of the induced permutation
  value of nodes adjacent to the node  $i$  and store the set into
   $L^*(v_i)$ ;
16:   Pad the length of  $L^*(v_i)$  to  $\max_{L^*}$ ;
17: END FOR
18: Output  $I^*, L^*$ .
```

5.1.3 Encrypt(k_4, m^*) $\rightarrow c$

数据拥有者对算法 BuildIndex 得到的节点数据项集合 m^* 中的每一项使用密钥 k_4 进行加密得到密文集合 c , 详细描述见算法 6。

算法 6. Encrypt(k_4, m^*) $\rightarrow c$

输入: 数据项 m^* 和密钥 k_4

输出: 密文集合 c

```

1: Initialize a set  $c$ ;
2: FOR  $j = 1:|m^*|$  DO
3:    $c_j \leftarrow \Phi.\text{Enc}_{k_4}(m_j^*)$ ;
4:   Store  $c_j$  into  $c$ ;
5: END FOR
```

6: Output c .

5.1.4 Token(K, w) $\rightarrow \tau$

算法的执行代理向数据拥有者发送自己感兴趣的关键词 w , 数据拥有者利用密钥 k_1, k_2 生成令牌 $\tau := (f_{k_1}(w), g_{k_2}(w))$ 返回给代理。

5.1.5 RunPSU(m_p, τ) $\rightarrow m_j \cup m_p$

运行代理与服务器之间的 PSU 协议, 详细描述如算法 RunPSU。代理首先调用 GenOKVS 算法, 输入数据集和密钥 k_3 , 输出向量 S ; 接着执行 Query 算法, 代理将向量 S 、令牌 τ 和随机串 s 发送到云服务器, 云服务器进行搜索、解码、解密等操作, 得到结果集 R ; 最后云服务器将 R 返回给代理, 代理执行 Decrypt 算法即可得到最终的并集结果。

算法 7. RunPSU(m_p, τ) $\rightarrow m_j \cup m_p$

输入: 数据集 m_p 和令牌 τ

输出: 并集 $m_j \cup m_p$

```

1:  $S \leftarrow \text{GenOKVS}(k_3, m_p)$ .
2:  $R \leftarrow \text{Query}(\tau, S)$ .
3:  $m_j \leftarrow \text{Decrypt}(k_4, R)$ .
4: Output  $m_j \cup m_p$ .
```

下面举一个具体的实例。我们假定数据拥有者拥有图 3(a)所示的社交网络图数据, 代理持有数据 m_6 和 m_7 , 并对关键词为 w_2 节点的所有邻居节点感兴趣。数据拥有者运行算法 GenKey、BuildIndex 和 Encrypt 后, 将加密数据上传至云端。代理运行 Token 算法, 向数据拥有者发送关键词 w_2 , 数据拥有者返回令牌 $\tau := (f_{k_1}(w_2), g_{k_2}(w_2))$ 。接下来运行代理与服务器间的 PSU 协议。首先运行 GenOKVS 算法。代理对自己持有节点标识符使用与数据拥有者相同的 π 置换生成 $\pi(6)$ 和 $\pi(7)$, 并生成随机串 s , 使用 k_3 进行两次加密得到 s_1, s_2 , 组成键值对 $(\pi(6), s_1)$ 和 $(\pi(7), s_2)$ 。代理调用 3H-GCT 的 Encode 算法对键值对 $(\pi(6), s_1)$ 和 $(\pi(7), s_2)$ 进行存储。接着运行 Query 算法。代理将令牌 τ 、随机串 s 和 3H-GCT 发送到云服务器进行查询, 根据令牌 τ 可以查询到关键词为 w_2 节点的所有邻居节点集合 L_{w_2} 为 $(\pi(1), \pi(4), \pi(5), \pi(6))$, 云服务器端使用 3H-GCT 算法的 Decode 算法依次对 $\pi(1)$ 、 $\pi(4)$ 、 $\pi(5)$ 和 $\pi(6)$

进行解码算法得到 $s_{\pi(1)}^*$ 、 $s_{\pi(4)}^*$ 、 $s_{\pi(5)}^*$ 和 $s_{\pi(6)}^*$ ，并使用 k_3 进行解密只有 $\Phi.\text{Dec}_{k_3}(s_{\pi(6)}^*) = s$ ，说明密文 $c_{\pi(6)}$ 属于交集元素，密文 $c_{\pi(1)}$ 、 $c_{\pi(4)}$ 、 $c_{\pi(5)}$ 属于云服务器独有的密文，故云服务器返回密文集合 $(c_{\pi(1)}$ 、 $c_{\pi(4)}$ 、 $c_{\pi(5)})$ 。最后运行 Decrypt 算法。代理使用密钥 k_4 解密即可得到节点数据项，从而完成了与云服务器之间的并集操作，降低了计算和通信开销。

5.2 安全性分析

本方案中云服务器被认为是“诚实且好奇”的，即会正确地执行查询任务并返回结果，但在此过程中会对敏感的数据感到好奇。根据 3.3 节中给出的泄露函数和安全模型，我们将对本文方案的安全性进行分析。

结构化加密具有关联性，支持加密复杂的数据类型，为了避免在搜索复杂数据类型过程中泄露信息，我们定义下面两个属性。第一个属性要求泄露函数 $\mathcal{L}_1$ 仅依赖于加密后的数据的结构 γ 和密文 c ，而不依赖于半保密数据项 v ，因为在查询过程中，会显示半保密数据项作为下一步查询的子令牌，若泄露依赖于半保密数据项，则会泄露更多与数据的结构有关的信息。第二个属性要求泄露的信息与查询顺序无关，因为子令牌是加密在索引中的，查询时子令牌将被显示，我们需要确保预先计算的子令牌与方案执行查询时显示的子令牌造成的泄露相同。下面我们形式化定义这两个属性。

定义 4(可链接性 Chainability). 一个结构化加密方案是可链接的需要满足针对自适应选择查询攻击 $(\mathcal{L}_1, \mathcal{L}_2)$ 安全，且泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 满足以下属性。

(1) 半保密数据独立性(semi-private independence): 存在泄露函数 $\mathcal{L}_1$ 使得 $\mathcal{L}_1(\delta, (m, v)) = \mathcal{L}_1(\delta, m)$ 。

(2) 顺序独立性(order independence): 存在一种变换 $\mathcal{Z}$ ，使得对于任何的数据的结构 δ 与任意一组查询序列 $\{q_1, \dots, q_t\}$ 及任意随机排列 p 都有： $\mathcal{Z}(\mathcal{L}_2(\delta, q_1), \dots, \mathcal{L}_2(\delta, q_t), p) = (\mathcal{L}_2(\delta, q_{p_1}), \dots, \mathcal{L}_2(\delta, q_{p_t}))$ 。

若本文方案满足以下等式：

$$\mathcal{L}_1(G, m, v, L^*, I^*) = (|R(w)|, |L_w|, |R|) \quad (2)$$

$$\mathcal{L}_2(G, w, L^*, I^*) = (\text{SP}(w), \text{AP}(w), \text{SP}(i), \text{AP}(i)) \quad (3)$$

其中， $\text{SP}(w)$ 揭示了当前的查询是否与先前的查询相同， $\text{AP}(w)$ 揭示了哪些相同的密文项被访问， $\text{SP}(i)$ 揭示了 $R(w)$ 的第 i 个节点是否也出现在之前的查询中， $\text{AP}(i)$ 揭示了查询结果是否与之前的查询结果含有共有的密文，那么本方案是针对自适应选择

查询攻击 $(\mathcal{L}_1, \mathcal{L}_2)$ 安全的。

定理 1. 本文提出的 STE-PSU 方案满足 CQA2 安全性。

证明. 首先构造模拟器 $\mathcal{S}$ ，给定泄露函数 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ ， $\mathcal{S}$ 构造索引 I' 。如果对于所有多项式时间的敌手都无法区分 Real 和 Ideal 两个游戏。那么本文的方案是 $(\mathcal{L}_1, \mathcal{L}_2)$ 针对自适应选择查询攻击安全的。其中 $\mathcal{A}$ 为敌手， $\mathcal{S}$ 为模拟器。

Game₀: 该游戏对应于真实实验 Real 的执行。挑战者通过 $K \leftarrow \text{GenKey}(1^\lambda)$ 生成密钥集合 $K = \{k_1, k_2, k_3, k_4\}$ 。敌手输出数据 (G, m, v) ，得到来自挑战者的索引表 I^* 、 L^* 和密文 c ，敌手发起一系列适应性的查询，对于每一个查询挑战者返回令牌 $\tau \leftarrow \text{Token}(K, w)$ 。最后由敌手返回一个 bit 位 b 。

Game₁: 与 **Game₀** 相似，除了在敌手输出数据 (G, m, v) 之后收到的数据变为 $\mathcal{S}(\mathcal{L}_1(G, m, v, I^*, L^*))$ ，并且对于敌手发起的每个查询，令牌 τ 计算如下： $R(w)$ 表示关键词为 w 的节点集合， v_w 表示关键词为 w 节点的半保密数据项集合，则 $\tau \leftarrow \mathcal{S}(\mathcal{L}_2(G, I^*, w), v_w)$ 。最后敌手输出一个 bit 位 b 。

假设存在多项式时间的敌手 $\mathcal{A}$ 使得 **Game₀** 和 **Game₁** 输出 1 的概率差异是不可忽略的，那么多项式时间的敌手 $\mathcal{A}$ 破坏了方案的 CQA2 安全性。

敌手 $\mathcal{A}$ 生成密钥集合 K ，并且接收到模拟器发送的 (G, m, v) ，且 $v = (v_1, \dots, v_n)$ 。敌手 $\mathcal{A}$ 输出 (G, m, v) 并接收到 (I^*, L^*, c) 。敌手 $\mathcal{A}$ 回答模拟器 $\mathcal{S}$ 的每个查询并且返回令牌 τ 。最终模拟器与敌手的输出结果是相同的。

需要注意的是，如果敌手 $\mathcal{A}$ 运行在真实实验 Real 中，那么模拟器 $\mathcal{S}$ 的输出结果与 **Game₀** 相同；若敌手 $\mathcal{A}$ 运行在理想实验 Ideal 中，那么模拟器 $\mathcal{S}$ 的输出结果与 **Game₁** 相同。

Game₂: 与 **Game₁** 相似，区别在于对令牌 τ 的计算，**Game₂** 只有需要时才计算。 τ 的计算如下：

$$\tau \leftarrow (\mathcal{L}_2(G, I^*, L^*, w), v_w), v_w := (v_i)_{i \in R(w)}$$

因为令牌的生成是无状态的，所以 **Game₂** 与 **Game₁** 等价。

Game₃: 与 **Game₂** 相似，区别在于 $\mathcal{L}_1(G, m, v, I^*, L^*)$ 与 $\mathcal{L}_2((G, I^*, L^*, w), v_w)$ 是直接提供的而不是通过 (G, m, v, w) 计算得到的。

观察可知，游戏 **Game₃** 相当于模拟器 $\mathcal{S}$ 执行理想实验 $\text{Ideal}_{\mathcal{A}, \mathcal{S}}(\lambda)$ 。显然，**Game₂** 和 **Game₃** 的输出分

布是相同的, 是不可区分的。

通过以上分析, 本文方案在执行过程中除了会泄露当前查询关键词之前是否被查询过, 哪些密文节点数据项被再次访问, 在检索邻居节点索引表过程中哪些节点之前被查询过, 没有其他信息泄露。本方案满足公式(3), 即针对自适应选择查询攻击是 $\mathcal{L}_2$ 安全的。另外, 服务器知道的数据包括加密的节点数据项、安全的索引结构以及查询过程中的半保密数据项。敌手 $\mathcal{A}$ 对关键词 w 进行查询, 可以获得包含 w 的节点数量 $|R(w)|$, $R(w)$ 中节点的邻居节点集大小 $|L_w|$ 以及最终返回给代理的结果集 R 的大小, 其满足公式(2)。因此本文提出的方案针对自适应选择查询攻击是 $\mathcal{L}_1$ 安全的。

对于所有多项式时间的敌手 $\mathcal{A}$ 都无法区分 Real 和 Ideal 两个实验。如公式(4)所示。

$$|\Pr[\text{Real}_{\mathcal{A}}(\lambda) = 1] - \Pr[\text{Ideal}_{\mathcal{A}, \mathcal{S}}(\lambda) = 1]| \leq \text{negl}(\lambda) \quad (4)$$

综上所述, 本文的 STE-PSU 方案是对于自适应选择查询攻击 ($\mathcal{L}_1$, $\mathcal{L}_2$) 安全的。

6 性能评估

下面对本文方案进行理论分析与实验分析。理论分析对我们的 STE-PSU 方案复杂度进行分析。实验在一台 Intel(R)Core(TM)i9-13900HX 2.20 GHz 和 16 GB RAM 的 Windows 系统机器上运行, 采用 Python 语言实现。索引中的伪随机函数使用 SHA-256 哈希实现, 3H-GCT 算法中的对称加密使用 LowMC^[39] 加密方案。实验数据集采用斯坦福网络分析项目^①中音乐媒体服务 Deezer 的真实社交网络数据集。本文使用 Deezer 中克罗地亚(Croatia)的数据集, 我们生成该数据集的子集用于实验分析, 详细信息如表 2 所示。

6.1 理论分析

我们将本文方案与其他具有代表性的两方 PSU 协议进行理论对比结果如表 3 所示, 对比的指标包括通信复杂度、时间复杂度和加密方案。其中 κ 和 λ 分别表示计算安全参数和统计安全参数, 我们考虑平衡数据集的情况, n 表示输入集合的大小。

文献[40]与文献[31]都通过 AHE 来实现安全的求并集操作, 文献[40]基于多项式的 AHE 造成了二次的通信和时间复杂度, 文献[31]则使用 AHE 对布隆过滤器进行加密, 虽然复杂度较低, 但 AHE 仍然导致了极低的效率(在文献[32]的实验表明其效率比

文献[31]提高了 7607×倍, 主要是因为文献[31]低效 AHE 加密)。文献[32]基于多项式实现了 RPMT 协议, 由于协议需要执行 n 次, 使用分桶技术进一步降低了计算和通信复杂度, 但协议的复杂度仍然较高。文献[41]和文献[33]的主要原语都是 oblivious switching network (OSN) 协议, 但他们的协议计算和通信复杂度都是超线性的。文献[6]和本文的 STE-PSU 协议都是基于高效的 3H-GCT 结构, 且均具有线性的通信和时间复杂度, 但本方案的优势在于, 我们通过 PSU 协议提高了在结构化加密图上的查询效率。

表 2 数据集
Table 2 Datasets

数据集	节点	边	关键词
Croatia	54573	498202	84
C ₁	10000	20143	79
C ₂	20000	71993	82
C ₃	30000	152371	83
C ₄	40000	267885	84
C ₅	50000	419231	84

表 3 方案比较
Table 3 Scheme comparison

方案	通信复杂度	时间复杂度	加密方案
文献[40]	$O(\kappa^3 \times n^2)$	$O(n^2)$	公钥加密
文献[32]	$O(n \times \log n \times \kappa)$	$O(n \times \log n \times \log n)$	对称加密
文献[31]	$O(n \times \kappa \times \lambda)$	$O(n \times \lambda)$	公钥加密
文献[41]	$O(n \times \log n \times \kappa)$	$O(n \times \log n)$	对称加密
文献[33]	$O(n \times \log n \times \kappa)$	$O(n \times \log n)$	对称加密
文献[6] ^②	$O(n \times \kappa)$	$O(n)$	对称加密
Ours	$O(n \times \kappa)$	$O(n)$	对称加密

6.2 实验分析

6.2.1 索引开销

构建索引的时间开销如图 5 所示。我们通过结构化加密为社交网络图构建关联性的加密索引表 I^* 和 L^* 。对 I^* 的构建过程需要对每一索引项进行填充, 使用伪随机置换 f 和伪随机函数 g 对关键词进行加密, 并对索引项中的节点生成其伪随机函数值链接到索引表 L^* ; 对 L^* 所有索引项中的节点使用诱导置换进行加密。索引的构建时间随着图规模的增加而逐渐增大, 在 C₁ 和 C₅ 数据集上, 索引构建时间分别为 2291 ms 和 14292 ms。索引的存储开销如图 6 所示,

① <https://snap.stanford.edu>.

② 文献[6]提出了两种 PSU 协议: 基于公钥加密和基于对称加密的 PSU 协议, 由于 STE-PSU 基于对称加密, 因此我们只与[6]中基于对称加密的方案进行对比。

存储开销随着图规模的增加而逐渐增大。在所有的数据集上 I^* 的存储开销都大于 L^* , 因为 I^* 的索引项都进行填充至最大长度, 并且每个节点需要生成一个链接到 L^* 的伪随机函数值, 造成了更大的存储开销。在 C_1 和 C_5 数据集上, I^* 、 L^* 的存储开销分别为 4056KB、1174KB 和 21430KB、9466KB。

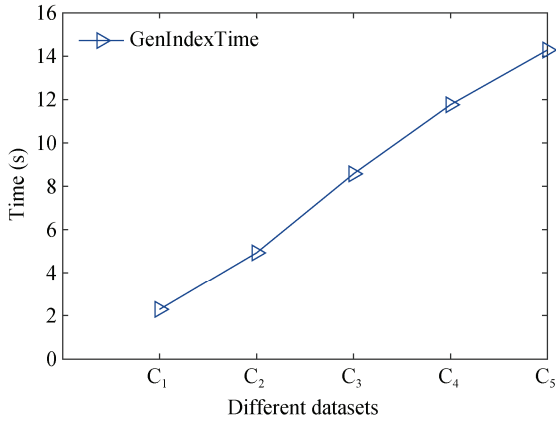


图 5 构建索引表的时间开销

Figure 5 The time of index construction

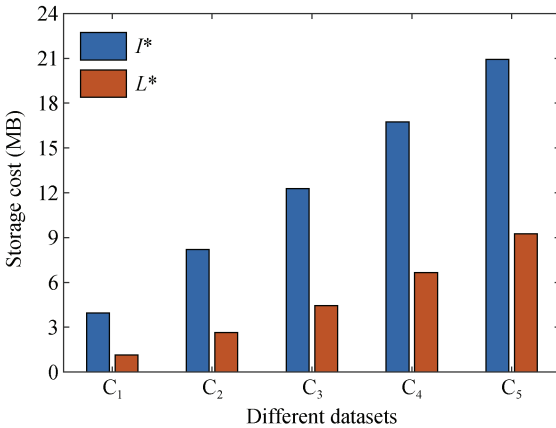


图 6 索引表的存储开销

Figure 6 The storage cost of index

6.2.2 STE-PSU 的效率

本文的 STE-PSU 协议在代理和服务器之间运行, 与传统两方 PSU 协议相比, 我们的 PSU 协议支持代理检索社交网络中与某一关键词节点相邻的所有节点集, 进而与代理本地数据集进行隐私集合求并, 因此协议的执行增加了检索加密索引表的过程, 但这并不会增加太多协议运行的时间。由于基于公钥操作的文献[40]与文献[31]协议的时间开销远高于我们的方案, 因此我们不与其进行实验对比。我们将 STE-PSU 与基于对称密钥的文献[32] (KRTW)、文献[41] (GMRSS)、文献[33] (JSZDG)、文献[6] (ZCL) 进行实验对比, 如图 7 所示。在我们的协议中, 代理向

服务器发起请求运行 PSU 协议时, 首先需要检索得到与关键词 w 节点相邻的节点集合 L_w , 我们通过进行多次实验使得 L_w 的大小 $|L_w|$ 略大于代理端节点数据集大小, 以使得我们能够测试出平衡数据集情况下协议的运行时间, 进而与对比方案进行实验对比。

图 7 的实验结果表明, 我们提出的 STE-PSU 协议比文献[32]、[41]、[33]中的方案效率更优, 与文献[6]中方案的效率相当, 这与我们理论分析的结果一致。实验也充分说明: 相比于基于多项式[32]的方案, STE-PSU 和文献[6]中的 OKVS 结构, 即 3H-GCT, 其效率更优; 同时, 实验也说明了 OSN 协议[33-41]的超线性复杂度会带来更多的开销。相比于文献[6], 本文通过执行 PSU 协议降低了在结构化加密社交网络图上查询的计算和通信开销。

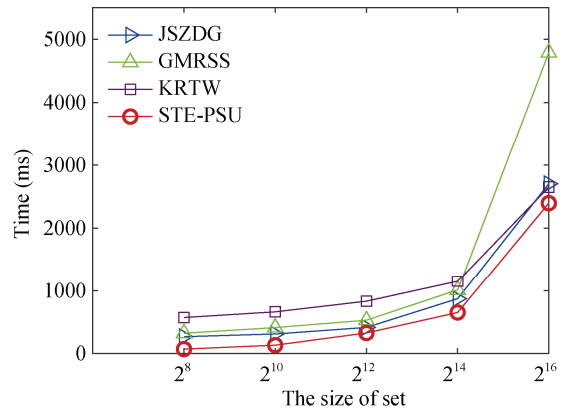


图 7 PSU 时间开销对比

Figure 7 The time comparison of PSU

此外, 我们还测试了 STE-PSU 方案各个阶段的时间开销如图 8 所示。其中 Encode 表示代理通过 3H-GCT 中 Encode 算法以代理端数据集为输入得到向量 S 的过程, Decode 表示服务器通过检索加密索引得到 L_w 后, 以 L_w 为输入进行 3H-GCT 中 Decode 算法的过程, $Dec s^*$ 表示对所有解码得到的 s^* 进行解密的过程。由此可知, STE-PSU 协议的运行效率由协议每个阶段的效率决定, OKVS 构造直接决定着编码与解码过程的效率, 对随机字符串的加密方案也对效率有很大影响。图 7 证明了我们方案中的 OKVS 构造 (3H-GCT) 以及对随机串使用对称加密方案 (LowMC) 共同决定了本方案的高效率。

7 总结

本文采用结构化加密将社交网络图加密后存储在云端, 通过代理与服务器之间高效且安全的 PSU 协议提高了搜索的效率和准确性, 降低了计算开销

和通信开销, 可用于丰富现实中代理的数据集。通过性能分析得出本方案是高效的, 并且通过安全性分析证明了本方案满足 CQA2 安全性要求。

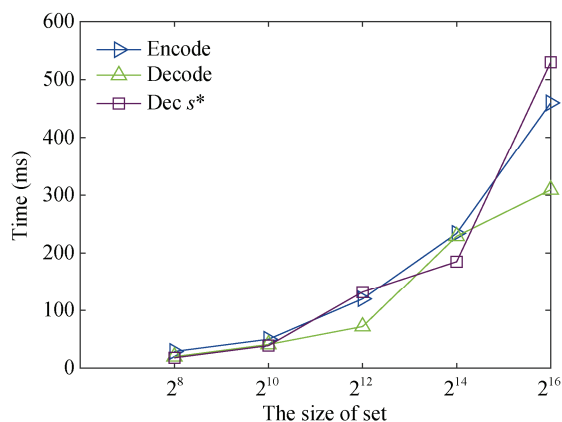


图 8 STE-PSU 各阶段的时间开销

Figure 8 The time for each stage of STE-PSU

下一步我们将考虑对于不同的结构化加密方案, 怎样利用现有的 OKVS 技术更好地降低其查询的计算开销与通信开销。同时我们考虑如何实现多方的基于结构化加密的 PSU 协议, 在保护隐私基础上, 提高搜索的效率, 实现更广泛的信息共享。

参考文献

- [1] Lai R W F, Chow S S M. Parallel and Dynamic Structured Encryption[M]. Security and Privacy in Communication Networks. ChamSpringer International Publishing2017: 219-238.
- [2] 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 2000*, 2000: 44-55.
- [3] Li F, Ma J F, Miao Y B, et al. A Survey on Searchable Symmetric Encryption[J]. *ACM Computing Surveys*, 2024, 56(5): 1-42.
- [4] Chase M, Kamara S. Structured Encryption and Controlled Disclosure[M]. *Advances in Cryptology - ASIACRYPT 2010*. Berlin, Heidelberg: Springer, 2010: 577-594.
- [5] Ali S, Shakeel M H, Khan I, et al. Predicting Attributes of Nodes Using Network Structure[J]. *ACM Transactions on Intelligent Systems and Technology*, 2021, 12(2): 1-23.
- [6] Zhang C, Chen Y, Liu W R, et al. Linear Private Set Union from Multi-Query Reverse Private Membership Test[C]. *The 32nd USENIX Conference on Security Symposium*, 2023: 337-354.
- [7] 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.
- [8] Kim K S, Kim M, Lee D, et al. Forward Secure Dynamic Searchable Symmetric Encryption with Efficient Updates[C]. *The 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017: 1449-1463.
- [9] Cash D, Jarecki S, Jutla C, et al. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries[M]. *Advances in Cryptology - CRYPTO 2013*. Berlin, HeidelbergSpringer2013: 353-373.
- [10] Du L L, Li K L, Liu Q, et al. Dynamic Multi-Client Searchable Symmetric Encryption with Support for Boolean Queries[J]. *Information Sciences*, 2020, 506: 234-257.
- [11] Liu X Q, Yang G M, Mu Y, et al. Multi-User Verifiable Searchable Symmetric Encryption for Cloud Storage[J]. *IEEE Transactions on Dependable and Secure Computing*, 2020, 17(6): 1322-1332.
- [12] Zhang Y, You L, Li Y. Tree-Based Public Key Encryption with Conjunctive Keyword Search[J]. *Security and Communication Networks*, 2021, 2021: 7034944.
- [13] Gan Q Q, Liu J K, Wang X M, et al. Verifiable Searchable Symmetric Encryption for Conjunctive Keyword Queries in Cloud Storage[J]. *Frontiers of Computer Science*, 2022, 16(6): 166820.
- [14] Zuo C, Sun S F, Liu J K, et al. Forward and Backward Private DSSE for Range Queries[J]. *IEEE Transactions on Dependable and Secure Computing*, 2022, 19(1): 328-338.
- [15] Zuo C, Sun S F, Liu J K, et al. Dynamic Searchable Symmetric Encryption with Forward and Stronger Backward Privacy[C]. *Computer Security - ESORICS 2019*, 2019: 283-303.
- [16] Zhao F, Peng C G, Xu D Q, et al. Attribute-Based Multi-User Collaborative Searchable Encryption in COVID-19[J]. *Computer Communications*, 2023, 205: 118-126.
- [17] He K, Chen J, Zhou Q X, et al. Secure Dynamic Searchable Symmetric Encryption with Constant Client Storage Cost[J]. *IEEE Transactions on Information Forensics and Security*, 2021, 16: 1538-1549.
- [18] Song Q Y, Liu Z T, Cao J H, et al. SAP-SSE: Protecting Search Patterns and Access Patterns in Searchable Symmetric Encryption[J]. *IEEE Transactions on Information Forensics and Security*, 2021, 16: 1795-1809.
- [19] Huang Y Y, Lv S Y, Liu Z L, et al. Cetus: An Efficient Symmetric Searchable Encryption Against File-Injection Attack with SGX[J]. *Science China Information Sciences*, 2021, 64(8): 182314.
- [20] Kamara S, Wei L. Garbled Circuits via Structured Encryption[M]. *Financial Cryptography and Data Security*. Berlin, HeidelbergSpringer2013: 177-188.
- [21] Xue Y J, Chen L X, Mu Y, et al. Structured Encryption for Knowledge Graphs[J]. *Information Sciences*, 2022, 605: 43-70.
- [22] 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.
- [23] Wang Q, Ren K, Du M X, et al. SecGDB: Graph Encryption for Exact Shortest Distance Queries with Efficient Updates[M]. *Financial Cryptography and Data Security*. ChamSpringer International Publishing2017: 79-97.
- [24] Du M X, Wu S K, Wang Q, et al. GraphShield: Dynamic Large Graphs for Secure Queries with Forward Privacy[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2022, 34(7): 3295-3308.
- [25] Wang W L, Jia Z L, Xu M F, et al. SPCS: Strong Privacy-Preserving-Constrained Shortest Distance Queries on Encrypted Graphs[J]. *IEEE Internet of Things Journal*, 2022, 9(22): 22516-22528.

- [26] Liang L, Lin J L, Qu Z, et al. SPG: Structure-Private Graph Database via SqueezePIR[J]. *Proceedings of the VLDB Endowment*, 2023, 16(7): 1615-1628.
- [27] Kolesnikov V, Kumaresan R, Rosulek M, et al. Efficient Batched Oblivious PRF with Applications to Private Set Intersection[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 818-829.
- [28] Pinkas B, Rosulek M, Trieu N, et al. SpOT-Light: Lightweight Private Set Intersection from Sparse OT Extension[M]. *Advances in Cryptology – CRYPTO 2019*. ChamSpringer International Publishing2019: 401-431.
- [29] Chase M, Miao P H. Private Set Intersection in the Internet Setting from Lightweight Oblivious PRF[C]. *Advances in Cryptology – CRYPTO 2020*, 2020: 34-63.
- [30] Ramanathan S, Mirkovic J, Yu M. Blag: Improving the accuracy of blacklists[C]. *NDSS*, 2020.
- [31] Davidson A, Cid C. An Efficient Toolkit for Computing Private Set Operations[M]. *Information Security and Privacy*. ChamSpringer International Publishing2017: 261-278.
- [32] Kolesnikov V, Rosulek M, Trieu N, et al. Scalable Private Set Union from Symmetric-Key Techniques[C]. *Advances in Cryptology – ASIACRYPT 2019*, 2019: 636-666.
- [33] Jia Y, Sun S F, Zhou H S, et al. Shuffle-based private set union: Faster and more secure[C]. *31st USENIX Security Symposium*, 2022: 2947-2964.
- [34] Bienstock A, Patel S, Seo J Y, et al. Near-Optimal Oblivious Key-Value Stores for Efficient PSI, PSU and Volume-Hiding Multi-Maps[C]. *The 32nd USENIX Conference on Security Symposium*, 2023: 301-318.
- [35] Garimella G, Pinkas B, Rosulek M, et al. Oblivious Key-Value Stores and Amplification for Private Set Intersection[C]. *Advances in Cryptology – CRYPTO 2021*, 2021: 395-425.
- [36] Pinkas B, Rosulek M, Trieu N, et al. PSI from PaXoS: Fast, Malicious Private Set Intersection[M]. *Advances in Cryptology – EUROCRYPT 2020*. ChamSpringer International Publishing2020: 739-767.
- [37] Dong C Y, Chen L Q, Wen Z K. When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol[C]. *The 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013: 789-800.
- [38] Walzer S. Peeling Close to the Orientability Threshold – Spatial Coupling in Hashing-Based Data Structures[M]. *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2021: 2194-2211.
- [39] Albrecht M R, Rechberger C, Schneider T, et al. Ciphers for MPC and FHE[M]. *Advances in Cryptology - EUROCRYPT 2015*. Berlin, Heidelberg: Springer, 2015: 430-454.
- [40] Kissner L, Song D. Privacy-Preserving Set Operations[M]. *Advances in Cryptology – CRYPTO 2005*. Berlin, Heidelberg: Springer, 2005: 241-257.
- [41] Garimella G, Mohassel P, Rosulek M, et al. Private Set Operations from Oblivious Switching[M]. *Public-Key Cryptography – PKC 2021*. ChamSpringer International Publishing2021: 591-617.



姜鹏 于 2021 年在池州学院计算机科学与技术专业获得学士学位。现在福建师范大学网络空间安全专业攻读硕士学位。研究领域为结构化加密、云存储安全、安全多方计算。Email: jiang_1813@163.com



陈兰香 (通信作者) 博士, 教授, 博士生导师, 中国计算机学会 (CCF) 会员, 主要研究兴趣包括密码学及其应用、隐私计算。Email: lxiangchen@gmail.com