密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2020, 7(1): 1–19

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

隐私集合计算中的关键数据结构研究*

作者一[1], 作者二[1,2], 作者三[1,2]

1. XX 大学 XXXX 实验室, 济南 250100
2. XXX 研究院, 北京 100190
通信作者: 作者一, E-mail: zuozhe1@net.cn

摘 要: 隐私集合计算 (private set operation, PSO) 是安全多方计算领域的热点问题, 它允许两个参与方对各自私有集合进行安全计算, 同时避免额外信息泄露. 常见的 PSO 协议包括隐私集合求交 (private set intersection, PSI) 和隐私集合求并 (private set union, PSU). PSO 协议的设计中往往运用了一些优化技巧降低协议的渐进复杂度. 除对 PSO 协议依赖的底层密码协议进行细节上的优化之外, 各种高级数据结构的应用也是重要优化技巧之一. PSO 协议涉及到不同的数据结构, 但各种数据结构的使用方式较为凌乱, 尚且缺乏梳理. 首先, 本文将现有 PSO 协议概括为三个阶段: 数据对齐、数据编码和成员测试, 并罗列了各阶段对应的数据结构, 分别是哈希表、不经意键值对存储和过滤器. 其次, 在明确了各类数据结构的定义和构造方式的基础上, 本文对比了不同数据结构在具体 PSO 协议中的应用场景, 并总结出了其应用方式. 最后, 本文为三类共 9 种数据结构提供了性能对比分析与基准测试.

关键词: 数据结构; 隐私集合计算

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000XXX

中文引用格式: 作者一, 作者二, 作者三. 隐私集合计算中的关键数据结构研究[J]. 密码学报, 2020, 7(1): 1–19.

英文引用格式: ZUO Z Y, ZUO Z E, ZUO Z S. A survey on key data structures for private set operation[J]. Journal of Cryptologic Research, 2020, 7(1): 1–19.

A Survey on Key Data Structures for Private Set Operation

ZUO Zhe-Yi[1], ZUO Zhe-Er[1,2], ZUO Zhe-San[1,2]

1. Lab of XXXX, XXXX University, Jinan 250100, China
2. Academy of XXXX, Beijing 100190, China
Corresponding author: ZUO Zhe-Yi, E-mail: zuozhe1@net.cn

Abstract: Private set operation (PSO) is a hot topic in the field of secure multi-party computation. It allows two parties to perform secure computations on their own private sets without additional information leakage. Genarally, PSO protocols include private set intersection (PSI) and private set union (PSU). PSO protocols are often designed with optimization techniques to reduce the asymptotic complexity. For example, optimizing the details of underlying cryptographic protocols. Becides, the application of various high-level data structures is also an important technique. PSO protocols involve different data structures, but the way they are used is messy and not yet organized. First, this

* 基金项目: 国家重点研发计划 (XXXX); 国家自然科学基金 (XXXX)
Foundation: National Key Research and Development Program of China (XXXX); National Natural Science Foundation of China (XXXX)
收稿日期: 2019-05-01 定稿日期: 2019-09-01

paper summarizes the existing PSO protocols into three phases: data alignment, data encoding and membership testing, and lists the data structures corresponding to each phase, namely, hashing tables, oblivious key-value stores and filters. Second, this paper compares the application scenarios of various data structures in specific PSO protocols and summarizes their different application rulls. Finally, this paper provides a comparative performance analysis and benchmark tests for the three types of 9 data structures.

Key words: data structure; private set operation

1 引言

大数据时代, 基于海量数据的计算和分析结果可以为科研、医疗、金融等领域提供更好的支持. 数据是机构或个人的核心资产. 出于隐私保护及利益的考虑内部数据通常不对外开放, 这阻碍了数据的流通. 数据孤岛现象普遍存在, 数据的价值无法体现. 如何在数据分布在各方的情况下保障数据和隐私安全, 同时发挥数据价值, 是当前亟待解决的问题. 自 1982 年姚期智 [8] 通过“百万富翁”问题引入安全多方计算 (secure multi-party computation, MPC) 概念以来, MPC 已成为解决数据流通过程中数据安全问题的关键技术之一.

隐私集合计算 (private set operation, PSO) 是安全多方计算领域的热点问题, 它允许两个参与方对各自私有集合进行安全计算, 同时避免额外信息泄露. 常见的 PSO 协议包括隐私集合求交 (private set intersection, PSI) 和隐私集合求并 (private set union, PSU). PSI 协议中, 发送方以集合 X 为输入, 接收方以集合 Y 为输入. 执行 PSI 协议之后, 发送方不获得任何信息, 接收方仅获得 $X \cap Y$ 的信息无法获得有关 $X \backslash Y$ 的任何信息. 与 PSI 协议相比, PSU 协议要求接收方只获得 $X \cup Y$ 的信息无法获得 $X \cap Y$ 的信息. 在隐私保护的场景中, PSO 具有重要意义, 如 PSI 的应用场景包括联系人匹配 [9]、计算广告转化率 [10]、DNA 检测与模式匹配 [11] 和接触者追踪 [12] 等; PSU 的应用场景包括利用 IP 黑名单联合查询进行网络风险评估 [13] 与隐私保护数据聚合 [14] 等.

PSO 协议的设计中往往运用了一些优化技巧降低协议的渐进复杂度. 除对 PSO 协议依赖的底层密码协议进行细节上的优化之外, 各类高级数据结构的应用也是重要优化技巧之一. PSO 协议涉及到不同的数据结构, 但各类数据结构的使用方式较为凌乱, 尚且缺乏梳理. 本文梳理了各类高级数据结构发挥的作用、总结它们在不同使用场景的应用方式, 可以为设计 PSO 协议、优化 PSO 方案提供很大的帮助. 已有的 PSO 协议大体上包括三个阶段: (1) 数据对齐, 将集合中的元素按照一定的规则划分并排列而非顺序排列; (2) 数据编码, 根据给定的键值对进行编码, 从而在一系列键和值之间建立映射关系; (3) 成员测试, 判断一个元素是否属于某一集合. 上述三个阶段对应的数据结构分别是哈希表、不经意键值对存储 (oblivious key-value store, OKVS) 和过滤器. 因此总体来看, 各类数据结构在 PSO 中发挥的主要作用概括如下.

数据对齐. 朴素的 PSO 协议中, 发送方需要对所有 $x \in X$ 测试其是否属于集合 Y. 因此当两方集合规模为 $O(n)$ 时协议的复杂度为 $O(n^2)$. 利用哈希表能够大幅降低协议的渐进复杂度. 具体做法为, 两方分别将各自集合中的元素映射到 m 个桶构成的哈希表, 该操作相当于将原始集合划分成了 m 个不相交的子集并且两方集合中相同元素会映射到相同索引的桶. 此外, 为防止额外信息泄露, 双方需要向桶中填充哑元 (dummy item). 哈希表实现了将集合中的元素按照一定的规则划分并排列, 原始集合的 PSO 协议等价于 m 个子集的 PSO 协议. PSO 协议中涉及到的哈希表类型包括简单哈希表 (simple hashing table) 和布谷鸟哈希表 (cuckoo hashing table) [15].

数据编码. 目前, 一些 PSO 协议依赖于具有不经意性质的数据结构对数据集编码. 在 Crypto 2021 上, Garimella 等人 [20] 将满足该性质的数据结构抽象成 OKVS. 对于给定的键值对 $\{k_i, v_i\}_{i\in[n]}$, OKVS 可以在大小为 m 的空间上存储映射关系 $k_i \mapsto v_i$. 不经意性表现为当 v_i 为随机值时, OKVS 可以隐藏其对应的键 k_i. 在 PSO 协议中, k_i 一般对应集合中的元素而 v_i 一般对应协议中的关键辅助信息. 例如, 发送方利用键值对 $\{x_i, v_i\}_{i\in[n]}$ 构造 OKVS, 接收方利用 $\{y_i\}_{i\in[n]}$ 作为键解码得到 $\{v_i^*\}_{i\in[n]}$. 两方可根据集合 $\{v_i\}_{i\in[n]}$ 和 $\{v_i^*\}_{i\in[n]}$ 求取交集或并集. PSO 协议中涉及到的 OKVS 类型包括多项式、乱码布隆过

滤器 (garbled bloom filter, GBF) [19] 和乱码布谷鸟哈希表 (garbled cuckoo table, GCT) [20,57,60].

成员测试. 为了减小两方交互过程中直接发送原始加密数据集造成的通信开销, PSO 协议可以利用过滤器对发送的数据集进行压缩. 一般情况下, 加密数据集中的密文长度为 128 比特 (AES 加密后的密文)、284 或 256 比特 (椭圆曲线上的点) 或者 2048 比特 (有限域中的元素或公钥加密后的密文) [9]. 因此使用过滤器可以极大的减小发送和存储的数据量. 具体做法为, 发送方利用待发送数据集合 X 构造过滤器并将过滤器发送给接收方, 接收方通过查询操作可获知查询元素 y 是否属于集合 X. PSO 协议中涉及到的过滤器类型包括布隆过滤器 (bloom filter) [16] 和布谷鸟过滤器 (cuckoo filter) [17]. 另外, 本文考虑使用压缩性能更好、时间效率更高的真空布隆过滤器 (vacuum filter) [18] 替代上述两种过滤器.

综上所述, PSO 是 MPC 的一个重要研究分支, 是近几年来国内外的研究热点. 各类高级数据结构的应用使得 PSO 协议的效率得到了极大的提高. 本文梳理和总结了各类数据结构发挥的作用并为其提供性能对比分析与基准测试, 可供 PSO 协议选择更高效更合适的优化技术.

2 预备知识

2.1 符号说明

本文中出现的符号和其描述说明如表1所示.

表 **1** 符号说明
Table 1 The notation and description

符号	描述
S, R	发送方和接收方
X, Y	发送方和接收方的集合
x_i, y_j	集合 X 和集合 Y 中的第 i 和第 j 个元素
n_x, n_y	发送方和接收方集合的大小, 大多数情况 $n_x = n_y = n$
m	数据结构中桶的个数
$[m]$	集合 $\{1, 2, \cdots, m\}$
b	每个桶允许包含的元素最大数量
λ	统计安全参数
γ	负载因子 ($\gamma = n/m$ 且 $0 \le \gamma \le 1$)

2.2 索引哈希的定义

本文中涉及的数据结构 (除多项式外) 的构造均利用了索引哈希计算元素映射在数据结构中的索引值.

定义 1 (索引哈希) 索引哈希表示为 $h : \{0,1\}^* \mapsto [m]$, 其输入为任意长度的字符串, 输出为 $[1, m]$ 之间的随机整数.

2.3 PSO 的定义及相关基础协议

2.3.1 PSO 的定义

本文中涉及的 PSO 协议包括隐私集合求交 (private set intersection, PSI), 隐私集合求并 (private set union, PSU).

定义 2 (PSO 协议) 协议中有两个参与方分别是发送方 S 和接收方 R, 其中 S 输入集合 $X = \{x_1, x_2, \cdots, x_{n_x}\}$, R 输入集合 $Y = \{y_1, y_2, \cdots, y_{n_y}\}$. 大多数情况下两方集合大小是公开的.

- PSI: S 不获得任何信息, R 获得 $X \cap Y$;
- PSU: S 不获得任何信息, R 获得 $X \cup Y$;

2.3.2 PSO 的相关基础协议

不经意传输

不经意传输协议 (oblivious transfer, OT) 是 MPC 中一个重要的密码组件, 可以用 OT 构建任何 MPC 协议 [67]. 姚氏乱码电路 [21]、GMW 协议 [22] 等通用 MPC 协议均大量应用 OT 协议. 以 PSO 为代表的高性能专用 MPC 协议也广泛应用 OT 协议. 在标准 2 选 1-OT 协议中, 发送方输入消息字符串 (m_0, m_1), 接收方输入选择比特 $r \in \{0,1\}$; 执行 OT 协议之后发送方不获得任何信息, 接收方仅获得其选择比特对应的字符串 m_r, 无法获知关于另一个消息 m_{1-r} 的任何信息. Impagliazzo 和 Rudich [68] 已证明, 无法通过黑盒调用对称密码学原语构造 OT 协议. 然而, 常见 MPC 协议大量应用 OT 协议, 直接使用基于公钥的 OT 协议将影响性能. 得益于高效的 OT 扩展协议 [23–25], 仅需要 $O(\kappa)$ 次公钥加密操作和 $O(n)$ 次快速的对称加密操作即可获得 $n \gg \kappa$ 个 OT 实例.

不经意伪随机函数

不经意伪随机函数 (oblivious pseudorandom function, OPRF) 是一个两方协议, 发送方拥有伪随机函数 F 的密钥 k, 接收方拥有输入 x; 执行协议之后发送方不获得任何信息, 接收方获得 $F_k(x)$. OPRF 协议的功能函数可以描述为 $\mathcal{F}_{\mathsf{OPRF}} : (k, x) \to (\perp, F_k(x))$. Freedman 等人 [32] 基于 Naor-Reingold 伪随机函数构造了 OPRF, 该构造需要幂指操作且需要的 OT 实例个数与 PRF 输入元素的长度线性相关. Camenisch 等人 [31] 基于盲签名构造了 OPRF 协议. Kolesnikov 等人 [33] 仅基于 OT 扩展构造了批处理 OPRF 协议使得计算效率得到了极大的提升. 此外, Casacuberta 等人 [66] 对近年来的 OPRF 协议做出了系统的综述, 根据构造 OPRF 协议底层的伪随机函数将 OPRF 分成了四个类别分别是 Naor-Reingold、Dodis-Yampolskiy、Hashed Diffie-Hellman 以及通用构造.

同态加密

同态加密是一种常用的实现隐私保护的工具, 其同态性是指对明文进行加法和 (或) 乘法运算与对密文进行相应运算结果等价. 同态加密根据同态性质可分为加法同态加密 (Paillier 加密 [26])、乘法同态加密 (RSA 加密 [27]、ElGamal 加密 [28]) 和全同态加密 (FHE [29]、BGV [30]). 同态加密方案通常包含有 4 个 PPT 时间的算法: $\mathsf{HE.KeyGen}$ 算法生成公钥和私钥, $\mathsf{HE.Enc}$ 算法加密明文, $\mathsf{HE.Dec}$ 算法解密密文, $\mathsf{HE.Eval}$ 算法对密文进行同态计算. 尽管目前全同态加密算法的效率不足以应用于实际场景, 但其支持直接密文计算的性质使得同态加密仍受到广泛关注.

2.4 安全模型

MPC 常用的安全模型有半诚实模型 (semi-honest model)、隐蔽模型 (covert model) 和恶意模型 (malicious model), PSO 协议主要设计半诚实模型与恶意模型.

- 半诚实模型 (semi-honest model): 半诚实模型也称为被动安全模型. 在半诚实模型中, 参与方遵循协议规则诚实地执行协议, 但是会尝试从其他参与方的输入或协议的中间计算结果中获得额外信息.
- 恶意模型 (malicious model): 恶意模型也称为主动安全模型. 在恶意模型中, 参与方会任意地偏离协议规则以破坏协议的安全性, 包括恶意篡改输入信息、拒绝参与协议、提前终止协议等.

3 PSO 中数据结构的分类

PSO 中的数据结构在不同类型协议中发挥的作用主要可以分为三类: (1) 利用哈希表实现数据对齐; (2) 利用过滤器实现成员测试; (3) 利用 OKVS 实现数据编码. 图1中展示了 PSO 中数据结构的分类情况.

4 哈希表

4.1 哈希表的定义

哈希表是一种能以键值对形式存储数据的数据结构. 它由插入算法、查询算法和删除算法构成, 且满足正确性这一性质.

定义 3 (哈希表) 令 $\mathcal{K}$ 表示键 (key) 的集合, $\mathcal{V}$ 表示值 (value) 的集合. 具体的插入算法、查询算法和删除算法如下所示.

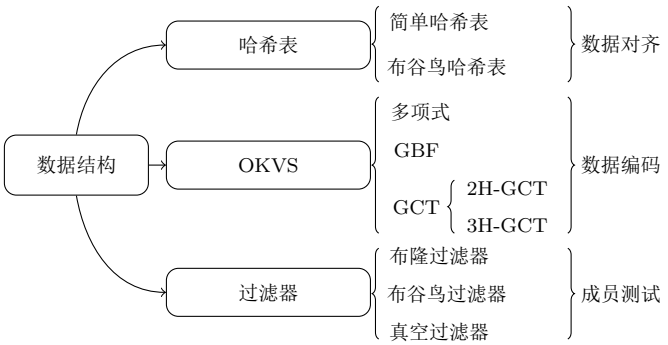


图 1 PSO 中数据结构的分类
Figure 1 Classification of data structures in PSO

- $\mathsf{Insert}(\{(k_1,v_1),\ldots,(k_n,v_n)\})$: 输入一组键值对 $\{(k_i,v_i)\}_{i\in[n]}\subseteq\mathcal{K}\times\mathcal{V}$, 输出构造的哈希表 D(或者以可忽略的概率输出错误指示符 $\perp$).
- $\mathsf{Find}(D,k)$: 输入构造的哈希表 D 和键 k, 若 $k\in\{k_i\}_{i\in[n]}$ 则输出 k 的映射值 v, 否则输出为空.
- $\mathsf{Remove}(D,k)$: 输入构造的哈希表 D 和键 k, 若 $k\in\{k_i\}_{i\in[n]}$ 则删除哈希表 D 中 k 的映射值 v, 否则不做任何操作.

正确性 (correctness). 对任意 $A\subseteq\mathcal{K}\times\mathcal{V}$ 都有: (1) 插入算法输出 $\perp$ 的概率是可忽略的; (2) 若 $\mathsf{Insert}(A)=D$ 且 $D\neq\perp$, 则对于任意 $(k,v)\in A$, $\mathsf{Find}(D,k)=v$.

4.2 哈希表的分类

简单哈希表

哈希表由 m 个桶 $\mathsf{B}[1],\mathsf{B}[2],\cdots,\mathsf{B}[m]$ 构成, 构建方式为利用一个索引哈希 h 将元素映射到表中. 元素 x 始终会被存放到桶 $\mathsf{B}[h(x)]$ 中, 不管该桶是否已经存放了其他元素. 文献 [44] 指出, 若将 n 个元素映射到 $m=n$ 个桶的哈希表中, 其中每个桶包含的元素最大数量 $b=O(\frac{\ln n}{\ln\ln n}(1+o(1)))$.

布谷鸟哈希表

Pagh 和 Rodler [15] 提出了布谷鸟哈希表, 构建方式为使用 $k\geq 2$ 个索引哈希 $h_1,h_2,\cdots,h_k$ 将元素映射到表中, 且需保证每个桶包含的元素最大数量 $b=1$. 插入元素 x 时, 首先计算映射位置 $h_1(x),h_2(x),\cdots,h_k(x)$. 如果 k 个映射位置存在“空闲”情况则直接插入元素 (如果多个映射位置都“空闲”则随机选择一个插入). 如果 k 个映射位置都已插入元素则随机选取桶 $\mathsf{B}[h_i(x)]$“逐出”其中的元素 x', 并插入元素 x. 对于元素 x', 将其插入到备选的 $k-1$ 个位置. 重复上述操作直到元素均插入哈希表或逐出次数达到了阈值. 出现后者情况时, 将无法插入的元素放入大小为 s 的额外空间 stash 中, 若 stash 已满仍有元素无法插入则说明此时表中元素已满, 无法再继续插入元素, 插入操作失败.

布谷鸟哈希表的构造失败概率受到三个参数的影响分别是额外空间 stash 的大小 s, 哈希函数的个数 k 和桶的个数 m. 令 $m=\epsilon n(\epsilon>1)$. 文献 [45] 指出对于任意常数 $c>0$ 和 $n\mapsto\infty$, 当 $k\geq 2\epsilon\ln\frac{1}{\epsilon-1}$ 且 $s\geq 0$ 时, 布谷鸟哈希表的构造失败概率为 $O(n^{1-c(s+1)})$. 大 O 符号中的常数通常不明确, 这使得在一组参数下很难计算出具体的失败概率. Pinkas 等人 [46] 投入大量精力进行实验 (验证失败概率上限为 p 需要运行超过 $1/p$ 次的实验) 并给出了当插入元素个数 $n=1024$, 失败概率分别为 2^{-30} 和 2^{-40}, $k\in\{2,3,4,5\}$, 且 $s=0$ 时需要的最小桶数 $m=\epsilon_{min}n$. 当失败概率为 $2^{-30}(2^{-40})$ 时, $k=2$ 时 $\epsilon_{min}=2.4$ ($\perp$); 当 $k=3$ 时 $\epsilon_{min}=1.2$ (1.27); $k=4$ 时 $\epsilon_{min}=1.07$ (1.09); $k=5$ 时 $\epsilon_{min}=1.04$ (1.05). 其中 $k=2$ 时, 该条件下无法保证失败概率为 2^{-40}.

针对上述哈希表的构造均存储元素本身的情况, 为进一步减少存储空间, Arbitman 等人 [47] 提出了置换哈希优化技巧. 该优化技巧可以减小哈希表中存储的元素长度. 主要思想为将元素 x 表示为比特形式并拆分为两部分, 其中前 $\log m$ 比特用于定义元素映射到哈希表中的索引值, 其余部分用于替代元素本身

存放于哈希表中. 具体来说, 将元素 x 表示为 $x_L||x_R$, 其中 x_L 部分的长度 $|x_L| = \log m$. 计算元素 x 在表中的索引值 $x_L \oplus h(x_R)$, 并将 x_R 存放在对应位置. 相比于上述哈希方案, 哈希表中存放的元素长度减小了 $\log m$ 比特. 因此当元素长度接近 $\log m$ 时, 采用置换哈希优化的哈希表在空间效率上有很大的提升.

4.3 哈希表在 PSO 中的应用

4.3.1 PSO 的数据对齐阶段

朴素的 PSO 协议中, 发送方需要对所有 $x \in X$ 测试其是否属于集合 Y. 因此当两方集合规模为 $O(n)$ 时协议的复杂度为 $O(n^2)$. 利用哈希表能够大幅降低协议的渐进复杂度. 本文将 PSO 中利用哈希表将集合中的元素进行划分并排列的过程概括为数据对齐阶段, 如图2所示.

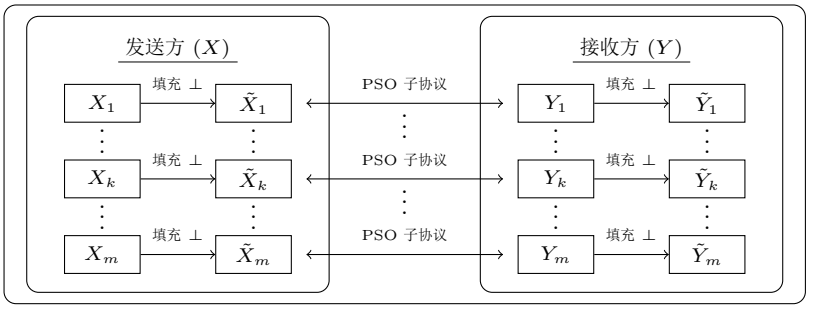


图 2 PSO 协议的数据对齐阶段

Figure 2 Data alignment phase of PSO protocols

发送方和接收方利用哈希表将集合 X 和 Y 分别划分成 m 个不相交的子集 $\{X_1, X_2, \cdots, X_m\}$ 和 $\{Y_1, Y_2, \cdots, Y_m\}$. 两方需使用相同的索引哈希构造哈希表, 保证两方集合中相同元素映射到相同索引的桶. 此外, 由于桶中元素的数量将暴露额外信息, 双方需要向桶中填充哑元使每个桶中的数量达到约定值 b. 发送方向桶 X_i 中填充一定数量的哑元得到 $\tilde{X}_i$, 同样接收方向桶 Y_i 中填充一定数量的哑元得到 $\tilde{Y}_i$. 双方在对齐后的子集 $(\tilde{X}_i, \tilde{Y}_i)$ 上执行 PSO 子协议.

4.3.2 哈希表的应用方式

哈希表在数据对齐阶段的应用方式主要可以分为两类, 其中若方案的计算开销或通信开销与集合中元素的长度相关, 则可以使用置换哈希压缩表中存放元素的长度优化哈希表的构造.

- 双方均使用简单哈希表. 该方式主要应用于根据集合中的元素编码为多项式这一范式设计的 PSO 协议 [38, 49]. 该类 PSO 协议中通常需要计算一个次数为 n 的高次多项式. 为了减小计算高次数多项式带来的开销, 发送方和接收方利用各自集合中的元素构建大小为 $m = O(n)$ 的哈希表, 每个桶中元素个数最多为 $b = O(\log n)$. 根据每个桶中的元素计算多项式只需计算一个次数为 b 的低次多项式. 因此, 通过构建哈希表, 实现了从计算一个 n 次多项式到计算 m 个 $O(\log n)$ 次多项式的转变, 且元素比较次数从 $O(n^2)$ 减小到 $O(n \log^2 n)$.
- 一方使用简单哈希表另一方使用布谷鸟哈希表. 该方式主要应用于 PSO 子协议中一方输入只能为单一点 (single point) 的情况, 如单点 OPRF 协议 [33, 44, 48]. 布谷鸟哈希表保证了每个桶中最多有一个元素. 一方使用布谷鸟哈希表将元素 x 存储到 $\{\mathsf{B}[h_1(x)], \mathsf{B}[h_2(x)], \cdots \mathsf{B}[h_k(x)]\}$ 其中一个桶中, 另一方使用简单哈希表将元素 y 存储到 k 个桶 $\{\mathsf{B}[h_1(y)], \mathsf{B}[h_2(y)], \cdots \mathsf{B}[h_k(y)]\}$. 逐桶进行比较, 若 $x = y$ 则无论 x 的存储位置是是 k 个位置中的哪一个都能得到正确结论. 将 n 个元素映射到 $m = O(n)$ 个桶的哈希表中时, 布谷鸟哈希表中每个桶最多拥有元素个数 $b = 1$, 简单哈希表中每个桶最多拥有元素个数 $b = O(\log n / \log \log n)$. 因此, 通过构建哈希表, 元素比较次数从 $O(n^2)$ 减小到 $O(n \log n / \log \log n)$.

4.4 哈希表在 PSO 中的研究进展

在 Eurocrypto 2004 上, Freedman 等人 [38] 首先将简单哈希表应用到了他们提出的 PSI 协议中. 在该协议中, 首先接收方利用乘法同态加密方案生成密钥对 $(\mathsf{pk}, \mathsf{sk})$, 将集合 Y 的元素作为多项式的根构建

n 次多项式 $P(z)=(y_1-z)(y_2-z)\cdots(y_n-z)=\sum_{i=0}^{n}a_iz^i$, 加密多项式系数发送给发送方. 发送方对集合 X 中的元素选择一个随机值 r_i, 利用同态加密方案计算 $r_i\cdot P(x_i)+x_i$ 并将其发送给接收方. 接收方解密, 如果元素 y_i 在交集中, 则对任意 r_i 都有 $r_i\cdot P(y_i)+y_i=y_i$; 否则 $r_i\cdot P(y_i)+y_i$ 是一个随机值. 为了减小对一个 n 次高阶多项式进行密态求值造成的开销, 该协议中给出了利用哈希表优化的方案. 两方均使用简单哈希表将集合中的元素对齐, 并逐桶执行 PSI 子协议.

随后出现了大量基于布谷鸟哈希表的 PSI 协议 [33,39,40,44,48], 这些协议的构造通常与 OT 扩展协议相结合, 其性能在相较于其他 PSI 协议具有极大优势. 其中, Kolesnikov 等人 [33] 在 CCS 2016 上提出的 PSI 协议为目前半诚实模型下计算开销最小、最具竞争力的 PSI 协议.

在 USENIX 2014 上, Pinkas 等人 [44] 首先将布谷鸟哈希表应用到了 PSI 协议中. 该协议基于 OPRF 协议构造, 关键思想在于将发送方和接收方集合中的元素替换为伪随机函数值, 其中发送方拥有密钥 k 可任意地计算其输入元素的伪随机函数值 $X'=\{F_k(x_i):x_i\in X\}$, 接收方无法自行计算只能得到 OPRF 协议中输出的伪随机函数值 $Y'=\{F_k(y_i):y_i\in Y\}$. 接收方通过对比 X' 和 Y' 可以得到交集. 该方案利用 2 选 1-OT 构造 OPRF 协议, 用函数表示为 $F((m_0,m_1),r)=m_r$, 其中 (m_0,m_1) 为 OPRF 的密钥, $r\in\{0,1\}$ 为 OPRF 的输入. 然而该 OPRF 协议为单点 OPRF 协议, 即该 OPRF 协议只允许接收方在单个点上计算 OPRF 值. 这导致双方需调用 $n\ell$ 次单点 OPRF 协议, ℓ 为集合中元素的长度, 接收方才能获得其所有集合元素的 OPRF 值. 这种情况与布谷鸟哈希表限制每个桶最多存放一个元素实现数据对齐的底层构造十分契合. 数据对齐后逐桶执行 PSI 子协议可大幅减小协议的开销. 观察到上述方案需要的 OT 实例个数不仅与集合中元素的个数相关还与元素的长度相关, Pinkas 等人 [48] 在 USENIX 2015 上提出使用置换哈希算法 [47] 优化布谷鸟哈希表的构造以减小每个桶中存放元素的长度. 置换哈希算法最早用于优化内存空间, 这是第一次应用到了密码学领域 [48]. 在 CCS 2016 上, Kolesnikov 等人 [33] 依然沿用了方案 [44] 中利用布谷鸟哈希表和单点 OPRF 协议构造 PSI 协议的范式, 提出了一个效率更高的方案. 他们利用伪随机编码 (pseudorandom code) 替换重复编码 (repetition code) 构造出了 2^ℓ 选 1-OT. 将 2 选 1-OT 替换为 2^ℓ 选 1-OT 构造 OPRF 协议, Kolesnikov 等人 [33] 的方案中 OT 实例的个数只与集合中元素的个数相关与元素长度无关.

在 CCS 2017 上, Chen 等人 [39] 基于布谷鸟哈希表和同态加密提出了一种适用于新型场景下的 PSI 协议, 即双方集合大小不对称 ($n_y\gg n_x$) 的非平衡场景. 与方案 [38] 类似, 接收方利用全同态加密方案生成密钥对 $(\mathsf{pk},\mathsf{sk})$, 加密集合 Y 中的元素得到密文 $\mathsf{HE.Enc}(y_1),\mathsf{HE.Enc}(y_2),\cdots,\mathsf{HE.Enc}(y_{n_y})$ 并发送给发送方. 发送方构建多项式 $P(z)=(x_1-z)(x_2-z)\cdots(x_{n_x}-z)=\sum_{i=0}^{n_x}a_iz^i$, 对于 $j\in[n_y]$ 发送方选择随机值 r_j, 计算 $r_j\cdot\prod_{i=0}^{n_x}\mathsf{HE.Enc}(a_i(y_j)^i)=r_j\cdot\mathsf{HE.Enc}(\sum_{i=0}^{n_x}a_i(y_j)^i)=r_j\cdot\mathsf{HE.Enc}(P(y_j))$ 并将其发送给接收方. 接收方解密, 如果 $y_j\in X$ 则对任意 r_j 有 $r_j\cdot P(y_j)=0$. 该方案将繁重的密态计算任务转移到了发送方, 拥有小集合的接收方只执行较轻的计算. 然而由于发送方需计算 $O(n_xn_y)$ 次同态乘法和加法, 该协议仍然不够高效. 结合布谷鸟哈希表与全同态加密算法的分窗、划分和模转换技术优化协议, 他们得到了一个通信开销为 $O(n_y\log n_x)$ 的高效 PSI 协议. 在 CCS 2018 上, Chen 等人 [40] 使用 OPRF 预处理阶段来实现比文献 [40] 更高的性能和安全性, 实现了针对恶意接收方的安全性. 方案 [39,40] 中均使用了置换哈希压缩布谷鸟哈希表中存放元素的长度以减小多项式插值的计算量. 此外, 与方案 [33,44,48] 不同的是, 方案 [39,40] 中的布谷鸟哈希表要求额外空间 stash 的大小 $s=0$, 否则无法与全同态加密方案兼容.

目前, 只有 Frikken [41] 和 Kolesnikov 等人 [49] 的 PSU 方案中应用了哈希表优化技术实现数据对齐. Frikken [41] 的 PSU 方案与 Freedman 等人 [38] 的 PSI 方案的构造类似, 均使用了多项式和同态加密且同样利用简单哈希表. Kolesnikov 等人 [49] 在 Asiacrypt 2019 上提出的 PSU 协议的核心为利用 OPRF 和多项式构造反向私有成员测试 (reverse private membership test, RPMT) 子协议. RPMT 协议可以测试发送方的元素是否属于接收方的集合, 并让接收方获得结果而发送方不获得任何信息. 由于他们构造的 RPMT 协议是一个单点协议发送发每次只能输入一个元素, 因此同样需要利用哈希表实现数据对齐. 然而, 该协议无法使用布谷鸟哈希表只能两方均使用简单哈希表. 具体原因为, 若发送方利用布谷鸟哈希将集合 X 映射到哈希表中, 当 $x\notin Y$ 时接收方可以获得发送方的元素 x 及其在布谷鸟哈希表中的索引. 由于元素 x 在布谷鸟哈希表中的索引取决于集合 X 中的所有元素, 上述情况可能导致发送方集合 X 信息泄露.

5 OKVS

5.1 OKVS 的定义

OKVS 是一种在一系列键和值之间建立映射关系的数据结构. 它由编码算法和解码算法构成, 且满足正确性和不经意性两个性质.

定义 4 (不经意键值对存储 [20]) 令 $\mathcal{K}$ 表示键 (key) 的集合, $\mathcal{V}$ 表示值 (value) 的集合. 具体的编码算法和解码算法如下所示.

- $\mathsf{Encode}(\{(k_1, v_1), \ldots, (k_n, v_n)\})$: 输入一组键值对 $\{(k_i, v_i)\}_{i\in[n]} \subseteq \mathcal{K} \times \mathcal{V}$, 输出 OKVS 编码结果 D(或者以可忽略的概率输出错误指示符 $\perp$).
- $\mathsf{Decode}(D, k)$: 输入 OKVS 编码结果 D 和需查询的键 k, 若 $k \in \{k_i\}_{i\in[n]}$ 则输出 k 的映射值 v, 否则输出随机值.

正确性 (correctness). 对任意 $A \subseteq \mathcal{K} \times \mathcal{V}$ 都有: (1) 编码算法输出 $\perp$ 的概率是可忽略的; (2) 若 $\mathsf{Encode}(A) = D$ 且 $D \neq \perp$, 则对于任意 $(k, v) \in A$, $\mathsf{Decode}(D, k) = v$.

不经意性 (obliviousness). 以不同的集合 $\mathcal{K}_1 = \{k_1^0, \ldots, k_n^0\}$ 和 $\mathcal{K}_2 = \{k_1^1, \ldots, k_n^1\}$ 为编码算法的输入, 并均匀随机地选择 $\{v_i\}_{i\in[n]} \leftarrow \mathcal{V}$, 若 $D \neq \perp$, 则分布 $\{D \mid v_i \leftarrow \mathcal{V}, i \in [n], \mathsf{Encode}((k_1^0, v_1), \ldots, (k_n^0, v_n))\}$ 与分布 $\{D \mid v_i \leftarrow \mathcal{V}, i \in [n], \mathsf{Encode}((k_1^1, v_1), \ldots, (k_n^1, v_n))\}$ 计算不可区分.

5.2 OKVS 的分类

下面描述了几种 OKVS 实例化, 并在表2中给出了其性能和特点对比.

表 2 元素个数为 n, 错误率为 $2^{-\lambda}$, 不同 OKVS 实例化的比较

Table 2 Comparison of various OKVS instantiations, for n items, with error $2^{-\lambda}$

OKVS	哈希函数的个数	负载因子	编码开销	解码开销
多项式	-	1	$O(n \log^2 n)$	$O(n \log^2 n)$
GBF	λ	$O(1/\lambda)$	$O(\lambda n)$	$O(\lambda n)$
2H-GCT	2	$0.4 - o(1)$	$O(\lambda n)$	$O(\lambda n)$
3H-GCT	3	$0.81 - o(1)$	$O(\lambda n)$	$O(\lambda n)$

多项式

多项式可看作一个最简单的 OKVS 实例化, 其中编码算法为利用 FFT 插值算法将键值对 $\{(x_i, y_i)\}_{i\in[n]}$ 构造为多项式 P, 其中 $P(x_i) = y_i$; 对应的解码算法为计算多项式关于输入 x 的值 $P(x)$. 利用多项式实例化 OKVS 数据结构的优势在于其负载因子为 1 达到了最优, 即空间开销最小. 然而, FFT 插值算法带来了较大的计算开销, 导致其编码开销和解码开销较大. 构造一个高效的 OKVS 实例化关键在于提升编码与解码效率的同时, 仅仅牺牲小部分的空间效率.

GBF

GBF 最早由 Dong 等人 [19] 在 2013 年提出. GBF 利用 k 个不同的索引哈希 $h_1, h_2, \cdots, h_k$ 将键值对 $\{(x_i, y_i)\}_{i\in[n]}$ 存储到 m 个桶中, 桶中允许包含的元素最大数量 $b = 1$. GBF 的编码过程为, 将键 x_i 映射的 k 个位置存放 λ 比特关于值 y_i 的秘密分享. 如果 k 个映射位置中已有字符串则共用该字符串, 剩下的位置通过异或得到共享字符串并填充入对应位置. 如果 k 个映射位置均已被占用则说明构造失败. 文献 [19] 指出, 当哈希函数个数 $k = \lambda$ 且桶个数 $m = 1.44\lambda n$ 时, GBF 的构造失败概率为 $2^{-\lambda}$. GBF 的解码过程为计算元素映射位置的异或值 $\mathsf{B}[h_1(x)] \oplus \mathsf{B}[h_2(x)] \cdots \oplus \mathsf{B}[h_k(x)]$.

GCT

GCT 的构造与 GBF 相似, 不同之处是 GBF 能以任意顺序编码待插入的键值对, 而 GCT 只能已特定顺序编码待插入的键值对. GCT 主要包括 2H-GCT、3H-GCT 和星型结构 (star architecture) 的 3H-GCT.

2H-GCT 由 Pinkas 等人 [57] 在 2020 年提出. 它的构造结合了布谷鸟哈希和 GBF 的思想, 即利用 2 个索引哈希而非 λ 个索引哈希构造 2H-GCT. 2H-GCT 的编码过程与 GBF 的编码过程相似只是算法触发终止的条件不同. GBF 编码过程中若元素映射的 k 个位置均已被占用则说明构造失败, 算法不进行额外处理. 然而当 $k=2$ 时该情况出现的概率较大, 2H-GCT 利用“剥离 (peeling)”操作处理该情况. 2H-GCT 中的剥离操作借助“布谷鸟图 (cuckoo graph)”完成. 布谷鸟图中 m 个节点表示为 $u_1,\cdots,u_m$, 键 k_i 和索引哈希 h_1,h_2 定义边 $(u_{h_1(k_i)},u_{h_2(k_i)})$. 剥离操作依次移除布谷鸟图中度为 1 的节点及其边. 该操作完成后若图中无剩余节点则按照移除节点的逆序依次为各节点赋值, 满足 $\mathsf{B}[h_1(k_i)]\oplus\mathsf{B}[h_2(k_i)]=v_i$. 若图中剩余节点, 说明形成了环 (cycle), 则优先为环中的节点赋值. 处理方法为将环中未赋值节点联立方程组, 利用高斯消元法求解. 2H-GCT 的解码过程与 GBF 相同, 计算元素映射位置的异或值 $\mathsf{B}[h_1(x)]\oplus\mathsf{B}[h_2(x)]$.

基于 3 个哈希函数构造的布谷鸟哈希表效率高于 2 个哈希函数构造的布谷鸟哈希表的事实, Garimella 等人 [20] 提出了 3H-GCT, 为目前最高效的 OKVS 实例化. 3H-GCT 的编码过程与解码过程均与 2H-GCT 相同. 类似布谷鸟哈希表在参数选择上面临的问题, 目前没有有效的方法可以分析出给定失败概率时, 2H-GCT 和 3H-GCT 需满足的具体参数. 基于上述问题, Garimella 等人给出了一种星型结构的 3H-GCT, 该结构可以将失败概率大 (如 2^{-15}) 的 3H-GCT 增强为失败概率小 (如 2^{-40}) 的 3H-GCT.

5.3 OKVS 在 PSO 中的应用

5.3.1 PSO 的数据编码阶段

目前, 一些 PSO 协议 [55,57,60] 依赖于具有不经意性质的数据结构对数据集编码. 在 Crypto 2021 上, Garimella 等人 [20] 将满足该性质的数据结构抽象成 OKVS. 对于给定的键值对 $\{k_i,v_i\}_{i\in[n]}$, OKVS 可以在大小为 m 的空间上存储映射关系 $k_i\mapsto v_i$. 不经意性表现为当 v_i 为随机值时, OKVS 可以隐藏其对应的键 k_i. 在 PSO 协议中, k_i 一般对应集合中的元素而 v_i 一般对应协议中的关键辅助信息. 本文将 PSO 中利用 OKVS 对一组键值对编码的过程概括为数据编码阶段, 如图3所示.

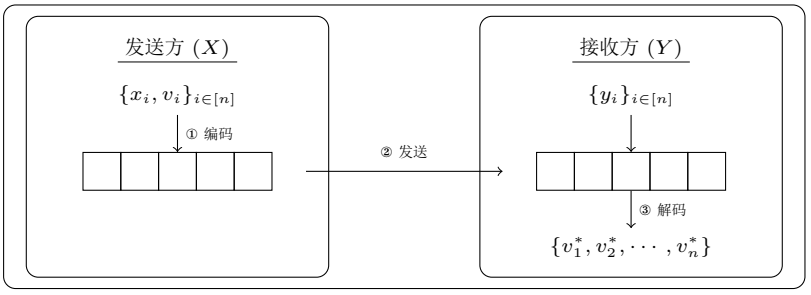


图 3 PSO 协议的数据编码阶段
Figure 3 Data encoding phase of PSO protocols

5.3.2 OKVS 的应用方式

数据编码阶段暗中实现了数据对齐. 如果 PSO 协议中应用了 OKVS(多项式除外), 则一般不再需要利用哈希表进行优化. 发送方 (或接收方) 利用键值对 $\{x_i,v_i\}_{i\in[n]}$ 构造 OKVS, 其中 x_i 表示其集合中的元素 v_i 表示随机值. 由于具有不经意性这一性质, OKVS 的编码结果可以直接发送给另一方而不会暴露集合中元素的信息. 接收方 (或发送方) 利用其集合元素 $\{y_i\}_{i\in[n]}$ 作为键解码得到 $\{v_i^*\}_{i\in[n]}$. 两方可根据集合 $\{v_i\}_{i\in[n]}$ 和 $\{v_i^*\}_{i\in[n]}$ 的信息求取交集或并集.

5.4 OKVS 在 PSO 中的研究进展

近年来, 大量 PSI 协议 [19,55–57,60] 基于 OKVS 实现. OKVS 已成为 PSI 协议设计中的一种重要工具, 然而其在 PSU 协议中的应用相对较少.

Dong 等人 [19] 在 CCS 2013 上提出了一个新的数据结构 GBF, 并构造了半诚实模型下的 PSI 协议. 该协议中发送方和接收方根据其各自集合中的元素分别生成一个 GBF 和布隆过滤器, 并逐桶执行消息长

度为 λ 的 OT 协议, 其中对于 $i \in [m]$ 第 i 个 OT 协议发送方的输入为 $\{\mathsf{GBF}[i], \perp\}$ 接收方的输入为 $\{\mathsf{BF}[i]\}$, 经过 m 个 OT 协议后接收方可获得关于两方交集的乱码布隆过滤器表示 $\mathsf{GBF}_{X\cap Y}$. 接收方通过查询自身集合中的元素是否属于 $\mathsf{GBF}_{X\cap Y}$ 获得交集信息. 其中, 发送方直接将 $\{x_i, x_i\}_{i\in[n]}$ 作为键值对构造 GBF, 这样构造的 GBF 并不具有不经意性. 在 CCS 2017 上, Kolesnikov 等人 [56] 才正式利用了 GBF 的不经意性实现数据编码, 构造了一个新的组件可编程的 OPRF 协议. 可编程的 OPRF 协议与 OPRF 协议类似, 不同之处在于发送方可在其输入的一组点 $\{x_i, y_i\}_{i\in[n]}$ 上对输出的 OPRF 值编程. 具体来说, 若接收方的输入 $x = x_i$ 则其得到的 OPRF 值 $F_k(x) = y_i$. 结合可编程的 OPRF 协议与零分享 (zero sharing), 他们得到了一个高效的多方 PSI 协议.

在 Eurocrypto 2020 上, Pinkas 等人 [57] 通过改进布谷鸟哈希算法构造了一个新的数据结构 2H-GCT, 并结合具有同态性质的 N 选 1-OT [61] 得到了一个恶意模型下的 PSI 协议. 接收方利用键值对 $\{y_i, H(y_i)\}_{i\in[n]}$ 构造 2H-GCT, 其中 H 表示随机预言机 (random oracle). 接收方将 2H-GCT 每个桶中的元素作为 OPRF 协议的输入, 计算其 OPRF 值. 而发送方得到 OPRF 的密钥可计算任意元素的 OPRF 值. 该协议并未利用 2H-GCT 的不经意性, 而是利用 2H-GCT 代替布谷鸟哈希表实现数据对齐. 2H-GCT 解决了布谷鸟哈希表构建恶意 PSI 时泄露发送方未在集合交集中的集合信息问题 [59]. 该协议的计算效率几乎与已知最快的半诚实模型下的 PSI 协议 [33] 一样高.

在 Crypto 2021 上, Garimella 等人 [20] 抽象出了 OKVS 数据结构. 他们指出之前的 PSI 协议中应用的多项式、GBF 和 2H-GCT 均满足 OKVS 的定义, 并且提出了一个更高效的 OKVS 实例化 3H-GCT. 若利用 3H-GCT 替换之前协议中的 OKVS 实例化可以直接得到一个更高效的协议. Rindal 和 Schoppmann [60] 在 Eurocrypto 2021 上利用 2H-GCT 与不经意向量线性求值 (vector oblivious linear evaluation, Vector-OLE) 构造了一个新的多点 OPRF 协议. 他们实现的 PSI 方案通信开销和计算开销均为 $O(n)$ 并且为目前通信开销最小的 PSI 方案, 在集合大小为 2^{20} 时, 该方案的通信开销为 53MB 比基于 Diffie-Hellman 的 PSI 协议 [63] 的通信开销更低.

6 过滤器

6.1 过滤器的定义

过滤器是一种空间效率高、可用于近似成员测试 (approximate membership query) 的数据结构. 过滤器利用微小的查询误判率换取空间效率. 查询误判率即过滤器对集合元素执行查询操作时做出错误判断的数量占总判断数量的比率. 它由插入算法、查询算法和删除算法构成, 且满足正确性这一性质.

定义 5 (过滤器) 令 $\mathcal{K}$ 表示元素的集合. 具体的插入算法、查询算法和删除算法如下所示.

- $\mathsf{Insert}(\{k_1, \ldots, k_n\})$: 输入一组元素 $\{k_i\}_{i\in[n]} \subseteq \mathcal{K}$, 输出构造的过滤器 D(或者以可忽略的概率输出错误指示符 $\perp$).
- $\mathsf{Find}(D, k)$: 输入构造的过滤器 D 和元素 k, 若 $k \in \{k_i\}_{i\in[n]}$ 则输出 1, 否则输出 0(由于存在查询误判率, 其输出 1 的概率为 $2^{-\lambda}$).
- $\mathsf{Remove}(D, k)$: 输入构造的过滤 D 和元素 k, 若 $k \in \{k_i\}_{i\in[n]}$ 则删除该元素, 否则不做任何操作.

正确性 (correctness). 对任意 $A \subseteq \mathcal{K}$ 都有: (1) 编码算法输出 $\perp$ 的概率是可忽略的; (2) 若 $\mathsf{Insert}(A) = D$ 且 $D \neq \perp$, 则对于任意 $x \in A$, $\mathsf{Find}(D, x) = 1$.

6.2 过滤器的分类

以下介绍了三种典型的过滤器, 分别是布隆过滤器、布谷鸟过滤器和真空过滤器. 表3描述了查询误判率为 $2^{-\lambda}$ 时三种过滤器的比较.

布隆过滤器

Bloom [16] 在 1970 年提出了布隆过滤器. 其主要构造为用 k 个索引哈希 $h_1, h_2, \cdots, h_k$ 将 n 个元素 $x_1, x_2, \cdots, x_n$ 映射到 m 个桶中. 桶中允许包含的元素最大数量 $b = 1$. 插入元素 x_i 时, 将映射的 k 个位置对应的桶 $\mathsf{B}[h_i(x_i)]$ 存放 1. 所有元素插入完成后未占用的桶中存放 0. 查询元素 x 时, 算法通过检查 k

表 3 查询误判率为 $2^{-\lambda}$, 不同过滤器的比较
Table 3 Comparison of various filters, with false positive rate $2^{-\lambda}$

过滤器	哈希函数的个数	负载因子	平均空间开销	是否支持删除	是否支持并行
布隆过滤器	λ	$1/(1.44\lambda)$	1.44λ	否	是
布谷鸟过滤器	2	γ	$(\lambda+3)/\gamma$	是	否
真空过滤器	2	γ	$(\lambda+3+\log_2(\gamma))/\gamma$	是	否

布谷鸟过滤器和真空过滤器每个桶中槽的数量 $b=4$.

个桶 $\mathsf{B}[h_i(x)]$ 中的值是否全 1 判断元素是否属于集合. 由于存在哈希碰撞使得布隆过滤器中出现多个元素用同一比特信息表示的情况, 标准布隆过滤器无法删除元素, 若需要删除某一个元素只能重新构建整个过滤器.

布谷鸟过滤器

2014 年 Fan 等人 [17] 提出了支持删除操作的布谷鸟过滤器. 布谷鸟过滤器利用两个索引哈希 h_1 和 h_2 将 n 个元素映射到 m 个桶中, 且每个桶中最多可以存放 b 个 ℓ 比特的指纹. 插入元素 x 时, 布谷鸟过滤器将关于元素 x 的指纹 f_x 存放在 $\mathsf{B}[i_1]$ 或 $\mathsf{B}[i_2]$ 中, 其中:

$$
\begin{aligned}
i_1 &= h_1(x) \\
i_2 &= i_1 \oplus h_2(f_x)
\end{aligned}
$$

在只知道元素的指纹和其中一个位置索引的情况下, 可以通过异或操作得到另一个位置的索引. 上述结论成立需基于桶个数 m 为 2 的幂次方这一假设, 否则通过异或操作计算得到的索引值可能会大于 m 造成溢出. 布谷鸟过滤器利用“逐出”操作 (与布谷鸟哈希表相同) 并设置逐出次数上限处理碰撞. 布谷鸟过滤器通过检查元素 x 映射到的两个可选桶 $\mathsf{B}[i_1]$ 和 $\mathsf{B}[i_2]$ 中是否存在指纹 f_x 判断其是否在集合中. 删除元素 x 通过删除布谷鸟过滤器中对应的指纹 f_x 实现.

真空过滤器

为了进一步提高空间效率, Wang 等人 [18] 在 2019 年提出了真空过滤器. 由于布谷鸟过滤器的构造基于桶个数 m 为 2 的幂次方这一假设, 在一定程度上会造成空间浪费. 例如, 构造布谷鸟过滤器时若实际上需要的桶的个数为 1025, m 需要设置为 2048, 造成将近 50% 的空间浪费. 为了解决这一问题, 真空过滤器将整个表划分成大小相同的块, 每个块中桶的个数 L 为 2 的幂次方, 并保证插入元素的两个可选桶 $\mathsf{B}[i_1]$ 和 $\mathsf{B}[i_2]$ 出现在同一个块中. 为了平衡负载因子 (load factor) 和空间访问效率 (locality), Wang 等人另外提出了将表划分成大小不同块的算法. 真空过滤器的插入策略和查询策略与布谷鸟过滤器相同, 且同样支持删除操作.

6.3 过滤器在 PSO 中的应用

6.3.1 PSO 的成员测试阶段

为了减小两方交互过程中直接发送原始加密数据集造成的通信开销, PSO 协议可以利用过滤器对发送的数据集进行压缩. 一般情况下, 加密数据集中的密文长度为 128 比特 (AES 加密后的密文)、284 或 256 比特 (椭圆曲线上的点) 或者 2048 比特 (有限域中的元素或公钥加密后的密文) [9]. 因此使用过滤器可以极大的减小发送和存储的数据量. 本文将 PSO 中利用过滤器判断某一元素是否属于某一集合的过程概括为成员测试阶段, 如图4所示.

6.3.2 过滤器的应用方式

由于过滤器不具有不经意性这一性质, 因此发送方无法将其集合元素以明文的形式插入一过滤器并直接将过滤器发送给接收方. 过滤器在成员测试阶段的应用方式主要分为以下两类:

- 数据集盲化后插入过滤器. 盲化过程可通过 Blind-RSA [53]、Diffie-Hellman 密钥交换 [63] 和 OPRF 协议 [33,44,48] 实现. 发送方将数据集 $\{x_i\}_{i\in[n]}$ 盲化后得到 $\{x_i^*\}_{i\in[n]}$ 并将盲化后的数据集插入过滤器. 盲化后的数据集不会暴露原始数据集, 因此得到的过滤器可以直接发送给接收方执行查询操

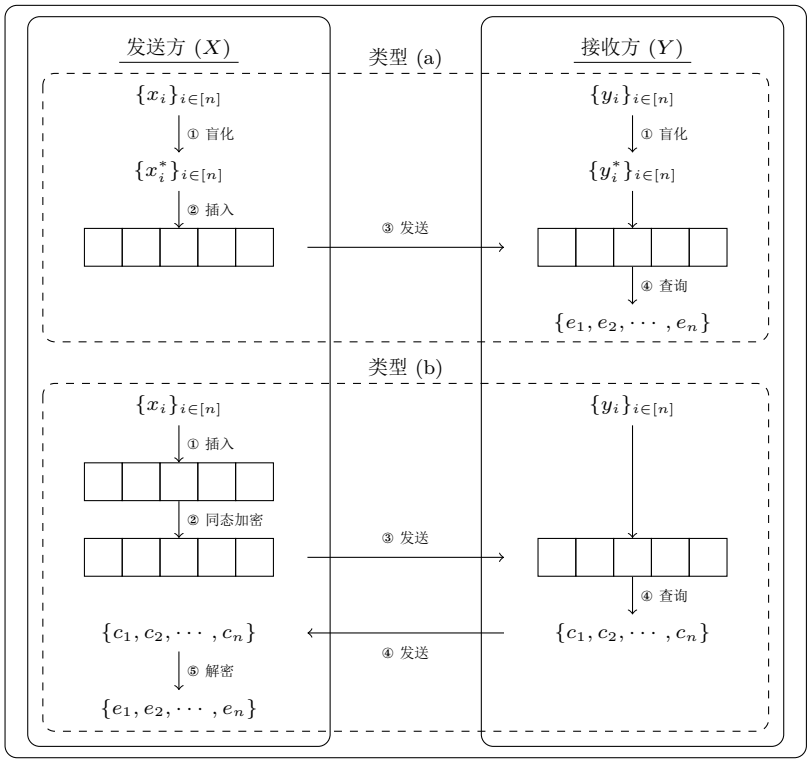


图 4 PSO 协议的成员测试阶段 (图中 e_i 表示 0 或 1, c_i 表示密文)
Figure 4 Membership test phase of PSO protocols(e_i denotes 0 or 1, c_i denotes ciphertext)

作. 接收方获得成员测试的结果 $\{e_1, e_2, \cdots, e_n\}$, 其中 $y_i \in X$ 时, $e_i = 1$; $y_i \notin X$ 时, $e_i = 0$. 具体内容见图4中的类型 (a).

- 数据集以明文的形式插入布隆过滤器, 并对得到的布隆过滤器加密. 该方式利用了布隆过滤器的特殊结构, 为布隆过滤器独有. 发送方将数据集 $\{x_i\}_{i\in[n]}$ 直接插入布隆过滤器并对其中的每一比特进行同态加密. 发送方将加密后的布隆过滤器发送给接收方, 接收方通过查询操作得到密文 $\{c_1, c_2, \cdots, c_n\}$. 若 $y_i \in X$ 则 $c_i = \mathsf{HE.Enc}(k)$, k 为哈希函数的个数 (或 $c_i = \mathsf{HE.Enc}(0)$, 取决于加密前的布隆过滤器是否逐比特反转), 否则 c_i 为对随机值的加密. 接收方将得到的密文发送给发送方, 发送方解密后得到成员测试的结果. 具体内容见图4中的类型 (b).

6.4 过滤器在 PSO 中的研究进展

在 PETS 2017 上, Kiss 等人 [9] 提出利用布隆过滤器压缩双方交互过程中的数据集减小通信开销. 他们将布隆过滤器应用到了现有的 PSI 协议中, 包括基于 Blind-RSA 的 PSI [53]、基于 Diffie-Hellman 密钥交换的 PSI [63]、基于 Naor-Reingold OPRF 的 PSI [34] 和基于乱码电路的 PSI [35].

在 Kiss 等人 [9] 的基础上, Resende 和 Aranha [52] 在 FC 2018 上首次将压缩性能更好的布谷鸟过滤器应用到了 PSI 方案中. 他们指出将布谷鸟过滤器与 Baldi 等人 [54] 的基于 OPRF 的 PSI 方案结合, 可以得到一个通信开销更小的优化方案. 文献 [54] 中的 OPRF 协议用函数可表示为 $F_\alpha(x) = H'(H(x)^\alpha)$, 其中 H 表示随机预言机, H' 表示将群元素映射到字符串的哈希函数. 具体来说, (1) 发送方输入集合 X 并选择密钥 $\alpha \in \mathbb{Z}_q^*$, 接收方输入集合 Y; (2) 接收方随机选择 $\beta_j \in \mathbb{Z}_q^*$ 并将 $H(y_j)^{\beta_j}$ 发送给发送方, 发送方返回 $(H(y_j)^{\beta_j})^\alpha$, 接收方将指数 β_j 移除并输出 $H'((H(y_j)^{\beta_j\alpha})^{1/\beta_j}) = H'(H(y_j)^\alpha)$; (3) 发送方计算 $H'(H(x_i)^\alpha)$ 并发送给接收方计算交集. 在优化的协议中, 发送方不直接发送盲化后的元素而是向接收方发送一个布谷鸟过滤器, 其中插入盲化后的元素 $H'(H(x_i)^\alpha)$; 接收方依次查询元素 $H'(H(y_j)^\alpha)$ 是否在过

滤器中. 利用布谷鸟过滤器将发送的元素集合进行压缩实现了通信开销的优化, 相比于 Baldi 等人 [54] 的方案, 该优化方案传输的数据量减少到原来的十分之三.

在 ACISP 2017 上, Davidson 和 Cid [51] 展示了利用布隆过滤器实现成员测试的一种特殊范式. 他们利用布隆过滤器与同态加密方案展示了一个 PSO 协议的设计框架. 在 PSI 协议中, 接收方创建其数据集的布隆过滤器, 将布隆过滤器逐比特反转, 并利用同态加密算法对布隆过滤器逐比特加密. 发送方对加密后的布隆过滤器执行查询操作获得查询结果的密文, 将密文发送给接收方. 接收方根据解密结果是否为 0 判断交集元素. 类似的, 他们描述了如何利用加密布隆过滤器实现 PSU 协议.

7 性能分析与测试

7.1 试验环境

本文的实验在 DELL OptiPlex 3060 上运行, 使用的是 Ubuntu 20.04.3 LTS 64 位操作系统, CPU 型号为 Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 内存大小为 16GB. 本文的实验采用 C++ 编程语言实现, 系统的 GCC 版本为 9.4.0(Ubuntu 9.4.0-5ubuntu1).

7.2 参数设置

本文对 PSO 中的关键数据结构提供了性能测试. 为保证所有测试均在同一基准下进行, 各类数据结构需选取不同的参数保证其失败概率 (查询误判率) 小于 2^{-40}. 涉及到的参数主要包括桶的个数 m, 哈希函数的个数 k, 额外储存空间 stash 的大小 s, 指纹长度 f, 每个桶允许包含的元素最大数量 b. 具体内容见表4. 由于多项式插值算法始终会成功, 失败概率为 0, 因此对多项式进行测试时不需要设置额外的参数. 星型结构的 3H-GCT 暂未公布开源代码.

表 **4** 参数说明
Table 4 Parameter Description

哈希表	简单哈希表	$m = 1.28n, k = 3$
	布谷鸟哈希表	$m = 1.28n, k = 3, s = 0$
OKVS	GBF	$m = 58n, k = 40$
	2H-GCT	$m = 2.4n$
	3H-GCT	$m = 1.3n$
过滤器	布隆过滤器	$m = 58n, k = 40$
	布谷鸟过滤器	$f = 16\ bits, b = 4$
	真空过滤器	$f = 16\ bits, b = 4$

7.3 测试结果

本文测试了元素个数 $n \in \{2^{12}, 2^{16}, 2^{20}\}$ 时各类数据结构的时间开销和空间开销. 由于利用哈希表实现数据对齐时不需要对哈希表执行查询操作, 且简单哈希表的开源代码中尚未提供查询操作接口, 本文没有对哈希表的查询操作提供测试. 其他类型数据结构的时间开销测试部分均包括了插入 (编码) 元素和查询 (解码) 元素测试.

表5中展示了哈希表的测试结果. 简单哈希表的插入性能由于布谷鸟哈希表. 原因在于, 简单哈希表不需要处理碰撞元素直接插入到映射位置, 而布谷鸟哈希表采用了“逐出”操作处理碰撞, 对插入元素的重定位会造成一定开销.

表6中展示了 OKVS 的测试结果. 多项式为空间性能最优的 OKVS, 负载因子达到了 1, 但其编码和解码开销较大. 其他基于哈希的 OKVS 包括 GBF、2H-GCT 和 3H-GCT 在编码和解码开销上得到了极大的提升, 均与元素个数线性相关. 最快的 OKVS 为 2H-GCT 和 3H-GCT, 它们的时间效率相当, 3H-GCT 的空间效率更优. 目前基于 3H-GCT 的 PSI 协议在时间开销和空间开销上都十分具有竞争力.

表7中展示了过滤器的测试结果. 三种过滤器均有较好的压缩性能. 布隆过滤器、布谷鸟过滤器和真空过滤器的时间效率和空间效率依次提升. 目前, PSO 中只考虑了利用布隆过滤器和布谷鸟过滤器实现成员测试, 如果使用压缩性能更好、时间效率更高的真空布隆过滤器替代上述两种过滤器, 现有方案可以直接得到优化.

表 **5** 元素个数为 n 且属于域 $\mathbb{F}_{2^{64}}$, 错误率为 2^{-40}, 不同哈希表性能测试
Table 5 Performance of various hashing tables, for n items from $\mathbb{F}_{2^{64}}$, with error 2^{-40}

哈希表/开源代码	$n = 2^{12}$		$n = 2^{16}$		$n = 2^{20}$	
	时间开销 (ms)	空间开销 (MB)	时间开销 (ms)	空间开销 (MB)	时间开销 (ms)	空间开销 (MB)
	插入	-	插入	-	插入	-
布谷鸟哈希表 [3]	0.2	0.32	5	5.19	166	84.27
简单哈希表 [3]	0.1	0.32	3	5.19	98	84.27

表 **6** 元素个数为 n 且属于域 $\mathbb{F}_{2^{64}}$, 错误率为 2^{-40}, 不同 OKVS 实例化性能测试
Table 6 Performance of various OKVS instantiations, for n items from $\mathbb{F}_{2^{64}}$, with error 2^{-40}

OKVS/开源代码	$n = 2^{12}$			$n = 2^{16}$			$n = 2^{20}$		
	时间开销 (ms)		空间开销 (MB)	时间开销 (ms)		空间开销 (MB)	时间开销 (ms)		空间开销 (MB)
	编码	解码	-	编码	解码	-	编码	解码	-
多项式 [6]	2808	2786	0.06	-	-	1.03	-	-	16.53
GBF [4]	18	2	14.41	296	89	230.43	4294	1292	3686.41
2H-GCT [5]	6	2	0.62	68	21	9.62	2232	549	163.29
3H-GCT [5]	5	1	0.34	100	23	5.63	3558	559	88.05
星型结构 [?]	-								

表 **7** 元素个数为 n 且属于域 $\mathbb{F}_{2^{64}}$, 查询误判率为 2^{-40}, 不同过滤器性能测试
Table 7 Performance of various filters, for n items from $\mathbb{F}_{2^{64}}$, with false positive probability 2^{-40}

过滤器/开源代码	$n = 2^{12}$			$n = 2^{16}$			$n = 2^{20}$		
	时间开销 (ms)		空间开销 (MB)	时间开销 (ms)		空间开销 (MB)	时间开销 (ms)		空间开销 (MB)
	插入	查询	-	插入	查询	-	插入	查询	-
布隆过滤器 [1]	2	0.2	0.03	51	3	0.45	338	47	7.19
布谷鸟过滤器 [2]	0.4	0.2	0.02	15	2	0.25	65	42	4
真空过滤器 [7]	0.1	0.1	0.02	1	1	0.16	33	18	2.13

8 总结与展望

随着安全多方计算研究的逐步深入, 隐私集合计算作为安全多方计算的一种重要应用近些年来效率得到了极大的提升. 其中各类数据结构的正确应用和高效实现发挥了重要作用. 本文总结了三类数据结构在隐私集合计算中的应用, 其中哈希表用于数据对齐; OKVS 用于数据编码; 过滤器用于成员测试. 另外对于哈希表、OKVS 和过滤器三类数据结构提供了性能对比分析和基准测试. 目前, PSO 中的关键数据结构在以下方面有待进一步的研究:

- 各类数据结构的相关开源代码均存在支持接口类型不丰富的问题, 例如对插入元素数据类型的支持, 序列化接口的实现以及删除操作接口的实现. 在 PSO 协议的设计中, 通常构造数据结构需要从一方

发送给另外一方. 序列化接口即将数据结构对象转换成字符串便于网络传输的实现. 只有 2H-GCT 和 3H-GCT 实现了序列化接口. 在 PSO 的应用场景中存在集合元素需要进行周期性更新的情况. 若数据结构支持删除操作, 则避免了集合元素更新时需要重新构建一个新的数据结构造成的计算开销问题. 哈希表、布谷鸟过滤器和真空过滤器实现删除操作比较直接, 直接删除元素或元素对应指纹即可; 而布隆过滤器和基于哈希的 OKVS(GBF, 2H-GCT, 3H-GCT, 星型结构) 实现删除操作需要在构造方法上进行优化. 其中 Fan 等人 [64] 和 Bonomi 等人 [65] 实现了支持删除操作的布隆过滤器.

- 在 PSO 协议的设计中, 大部分应用了过滤器和 OKVS 的方案均可以通过直接替换一个性能更好的过滤器或 OKVS 实现优化, 因此可以考虑利用真空过滤器和 3H-GCT 对现有方案进行优化. 此外, Rindal 和 Raghuraman [69] 提出了一个更高效的 OKVS.
- OKVS 已成为 PSI 协议设计中的一种重要工具, 大量高效的 PSI 协议基于 OKVS 实现, 然而其在 PSU 协议中的应用相对较少. 因此, OKVS 在 PSU 中的应用有待进一步研究.

参考文献

[1] Bloomfilter. https://github.com/ArashPartow/bloom

[2] Cuckoofilter. https://github.com/efficient/cuckoofilter

[3] libpsi. https://github.com/osu-crypto/libPSI

[4] MultipartyPSI. https://github.com/osu-crypto/MultipartyPSI

[5] Obliviousdictionary. https://github.com/cryptobiu/ObliviousDictionary

[6] SpOT-PSI. https://github.com/osu-crypto/SpOT-PSI

[7] Vacuumfilter. https://github.com/wuwuz/Vacuum-Filter

[8] YAO A C. Protocols for Secure Computations (Extended Abstract)[C]. 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. [S.l.]: IEEE Computer Society, 1982: 160–164. [DOI: 10.1109/SFCS.1982.38].

[9] KISS Á, LIU J, SCHNEIDER T, et al. Private Set Intersection for Unequal Set Sizes with Mobile Applications[J]. Proc. Priv. Enhancing Technol., 2017, 2017(4): 177–197. [DOI: 10.1515/popets-2017-0044].

[10] ION M, KREUTER B, NERGIZ A E, et al. On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality[C] . IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020. [S.l.]: IEEE, 2020: 370–389. [DOI: 10.1109/EuroSP48549.2020.00031].

[11] TRONCOSO-PASTORIZA J R, KATZENBEISSER S, CELIK M U. Privacy preserving error resilient dna searching through oblivious automata[C]. Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. [S.l.]: ACM, 2007: 519–528. [DOI: 10.1145/1315245.1315309].

[12] DUONG T, PHAN D H, TRIEU N. Catalic: Delegated PSI Cardinality with Applications to Contact Tracing[C]. Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III. [S.l.]: Springer, 2020: 870–899. [DOI: 10.1007/978-3-030-64840-4_29].

[13] HOGAN K, LUTHER N, SCHEAR N, et al. Secure Multiparty Computation for Cooperative Cyber Risk Assessment[C] . IEEE Cybersecurity Development, SecDev 2016, Boston, MA, USA, November 3-4, 2016. [S.l.]: IEEE Computer Society, 2016: 75–76. [DOI: 10.1109/SecDev.2016.028].

[14] BURKHART M, STRASSER M, MANY D, et al. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics[C]. 19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings. [S.l.]: USENIX Association, 2010: 223–240.

[15] PAGH R, RODLER F F. Cuckoo hashing[J]. J. Algorithms, 2004, 51(2): 122–144. [DOI: 10.1016/j.jalgor.2003.12.002].

[16] BLOOM B H. Space/Time Trade-offs in Hash Coding with Allowable Errors[J]. Commun. ACM, 1970, 13(7): 422–426. [DOI: 10.1145/362686.362692].

[17] FAN B, ANDERSEN D G, KAMINSKY M, et al. Cuckoo Filter: Practically Better Than Bloom[C]. Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, CoNEXT 2014, Sydney, Australia, December 2-5, 2014. [S.l.]: ACM, 2014: 75–88. [DOI: 10.1145/2674005.2674994].

[18] WANG M, ZHOU M, SHI S, et al. Vacuum Filters: More Space-Efficient and Faster Replacement for Bloom and Cuckoo Filters[J]. Proc. VLDB Endow., 2019, 13(2): 197–210.

[19] DONG C, CHEN L, WEN Z. When private set intersection meets big data: an efficient and scalable protocol[C].

2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013. [S.l.]: ACM, 2013: 789–800. [DOI: 10.1145/2508859.2516701].

[20] GARIMELLA G, PINKAS B, ROSULEK M, et al. Oblivious Key-Value Stores and Amplification for Private Set Intersection[C]. Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II. [S.l.]: Springer, 2021: 395–425. [DOI: 10.1007/978-3-030-84245-1_14].

[21] YAO A C. How to Generate and Exchange Secrets (Extended Abstract)[C]. 27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986. [S.l.]: IEEE Computer Society, 1986: 162–167. [DOI: 10.1109/SFCS.1986.25].

[22] GOLDREICH O, MICALI S, WIGDERSON A. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority[C]. Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. [S.l.]: ACM, 1987: 218–229. [DOI: 10.1145/28395.28420].

[23] ISHAI Y, KILIAN J, NISSIM K, et al. Extending Oblivious Transfers Efficiently[C]. Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. [S.l.]: Springer, 2003: 145–161. [DOI: 10.1007/978-3-540-45146-4_9].

[24] ASHAROV G, LINDELL Y, SCHNEIDER T, et al. More efficient oblivious transfer and extensions for faster secure computation[C]. 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013. [S.l.]: ACM, 2013: 535–548. [DOI: 10.1145/2508859.2516738].

[25] KOLESNIKOV V, KUMARESAN R. Improved OT Extension for Transferring Short Secrets[C]. Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. [S.l.]: Springer, 2013: 54–70. [DOI: 10.1007/978-3-642-40084-1_4].

[26] PAILLIER P. Public-key cryptosystems based on composite degree residuosity classes[C]. Proc of the 18th Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 1999: 223- 238. [DOI: 10.1007/3-540-48910-X_16]

[27] RIVEST R, SHAMIR A, ADLEMAN L. A method for obtaining digital signatures and public-key cryptosystems[J]. Communications of the ACM, 1978, 21(2): 120-126. [DOI: 10.1145/359340.359342]

[28] ELGAMAL T. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE Transactions on Information Theory, 1985, 31(4): 469-472. [DOI: 10.1109/TIT.1985.1057074]

[29] GENTRY C. A fully homomorphic encryption scheme[D]. Stanford, CA: Stanford University. 2009.

[30] BRAKERSKI Z, GENTRY C, VAIKUNTANATHAN V. (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) 6, 3 (2014), 13. [DOI: 10.1145/2633600]

[31] CAMENISCH J, NEVEN G, SHELAT A. Simulatable Adaptive Oblivious Transfer[C]. Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings. [S.l.]: Springer, 2007: 573–590. [DOI: 10.1007/978-3-540-72540-4_33].

[32] FREEDMAN M J, ISHAI Y, PINKAS B, et al. Keyword Search and Oblivious Pseudorandom Functions[C]. Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings. [S.l.]: Springer, 2005: 303–324. [DOI: 10.1007/978-3-540-30576-7_17].

[33] KOLESNIKOV V, KUMARESAN R, ROSULEK M, et al. Efficient Batched Oblivious PRF with Applications to Private Set Intersection[C]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. [S.l.]: ACM, 2016: 818–829. [DOI: 10.1145/2976749.2978381].

[34] NAOR M, REINGOLD O. Number-theoretic constructions of efficient pseudo-random functions[J]. ACM, 51(2):231–262, 2004. [DOI: 10.1145/972639.972643]

[35] PINKAS B, SCHEIDER T, SMART P N, et al. Secure two-party computation is practical[C]. In Advances in Cryptology –ASIACRYPT' 09, volume 5912 of LNCS, pages 250–267. Springer, 2009. [DOI: 10.1007/978-3-642-10366-7_15]

[36] NAOR M, PINKAS B. Oblivious Transfer and Polynomial Evaluation[C]. Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA. [S.l.]: ACM, 1999: 245–254. [DOI: 10.1145/301250.301312].

[37] GILBOA N. Two Party RSA Key Generation[C]. Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. [S.l.]: Springer, 1999: 116–129. [DOI: 10.1007/3-540-48405-1_8].

[38] FREEDMAN M J, NISSIM K, PINKAS B. Efficient Private Matching and Set Intersection[C]. Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. [S.l.]: Springer, 2004: 1–19. [DOI: 10.1007/978-

3-540-24676-3_1].

[39] CHEN H, LAINE K, RINDAL P. Fast Private Set Intersection from Homomorphic Encryption[C]. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. [S.l.]: ACM, 2017: 1243–1255. [DOI: 10.1145/3133956.3134061].

[40] CHEN H, HUANG Z, LAINE K, et al. Labeled PSI from Fully Homomorphic Encryption with Malicious Security[C]. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. [S.l.]: ACM, 2018: 1223–1237. [DOI: 10.1145/3243734.3243836].

[41] FRIKKEN B K. Privacy-preserving set union. In Jonathan Katz and Moti Yung, editors, ACNS 07, volume 4521 of LNCS, pages 237–252. Springer, Heidelberg, June 2007. [DOI: 10.1007/978-3-540-72738-5_16]

[42] HAZAY C. Oblivious Polynomial Evaluation and Secure Set-Intersection from Algebraic PRFs[J]. J. Cryptol., 2018, 31(2): 537–586. [DOI: 10.1007/s00145-017-9263-y].

[43] MITZENMACHER M. The Power of Two Choices in Randomized Load Balancing[J]. IEEE Trans. Parallel Distributed Syst., 2001, 12(10): 1094–1104. [DOI: 10.1109/71.963420].

[44] PINKAS B, SCHNEIDER T, ZOHNER M. Faster Private Set Intersection Based on OT Extension[C]. Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014. [S.l.]: USENIX Association, 2014: 797–812.

[45] KIRSCH A, MITZENMACHER M, WIEDER U. More Robust Hashing: Cuckoo Hashing with a Stash[J]. SIAM J. Comput., 2009, 39(4): 1543–1561. [DOI: 10.1137/080728743].

[46] PINKAS B, SCHNEIDER T, ZOHNER M. Scalable Private Set Intersection Based on OT Extension[J]. ACM Trans. Priv. Secur., 2018, 21(2): 7:1–7:35. [DOI: 10.1145/3154794].

[47] ARBITMAN Y, NAOR M, SEGEV G. Backyard Cuckoo Hashing: Constant Worst-Case Operations with a Succinct Representation[C]. 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA. [S.l.]: IEEE Computer Society, 2010: 787–796. [DOI: 10.1109/FOCS.2010.80].

[48] PINKAS B, SCHNEIDER T, SEGEV G, et al. Phasing: Private Set Intersection Using Permutation-based Hashing[C]. 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015. [S.l.]: USENIX Association, 2015: 515–530.

[49] KOLESNIKOV V, ROSULEK M, TRIEU N, et al. Scalable Private Set Union from Symmetric-Key Techniques[C]. Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II. [S.l.]: Springer, 2019: 636–666. [DOI: 10.1007/978-3-030-34621-8_23].

[50] RINDAL P, ROSULEK M. Improved Private Set Intersection Against Malicious Adversaries[C]. Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. 2017: 235–259. [DOI: 10.1007/978-3-319-56620-7_9].

[51] DAVIDSON A, CID C. An Efficient Toolkit for Computing Private Set Operations[C]. Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II. [S.l.]: Springer, 2017: 261–278. [DOI: 10.1007/978-3-319-59870-3_15].

[52] RESENDE A C D, ARANHA D F. Faster Unbalanced Private Set Intersection[C]. Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26 - March 2, 2018, Revised Selected Papers. [S.l.]: Springer, 2018: 203–221. [DOI: 10.1007/978-3-662-58387-6_11].

[53] CRISTOFARO D E, TSUDIK G. Practical private set intersection protocols with linear complexity. In Financial Cryptography and Data Security (FC' 10), volume 6052 of LNCS, pages 143–159. Springer, 2010. [DOI: 10.1007/978-3-642-14577-3_13]

[54] BALDI P, BARONIO R, CRISTOFARO E D, et al. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes[C]. Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011. [S.l.]: ACM, 2011: 691–702. [DOI: 10.1145/2046707.2046785].

[55] PINKAS B, ROSULEK M, TRIEU N, et al. Spot-light: Lightweight private set intersection from sparse ot extension[C]. Advances in Cryptology - CRYPTO 2021 - 39th Annual International Cryptology Conference, CRYPTO 2019, Berlin: Springer, 2019: 401-431. [DOI: 10.1007/978-3-030-26954-8_13]

[56] KOLESNIKOV V, MATANIA N, PINKAS B, et al. Practical Multi-party Private Set Intersection from Symmetric-Key[C]. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017: 1257-1272. [DOI: 10.1145/3133956.3134065]

[57] PINKAS B, ROSULEK M, TRIEU N, et al. PSI from PaXoS: Fast, Malicious Private Set Intersection[C]. Advances

in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. [S.l.]: Springer, 2020: 739–767. [DOI: 10.1007/978-3-030-45724-2_25].

[58] SHEN L Y, CHEN X J, SHI J Q, et al. Survey on Private Preserving Set Intersection Technology[J]. Journal of Computer Research and Development, 2017, 54(10): 2153-2169. [DOI: 10.7544/issn1000-1239.2017.20170461]
申立艳, 陈小军, 时金桥, 等. 隐私保护集合交集计算技术研究综述 [J]. 计算机研究与发展, 2017, 54(10): 2153-2169. [DOI: 10.7544/issn1000-1239.2017.20170461]

[59] WEI L F, LIU J H, ZHANG L, et al. Survey of Privacy Preserving Oriented Set Intersection Computation[J/OL]. Journal of Computer Research and Development, 2022, 59(8): 1782-1799. [DOI: 10.7544/issn1000-1239.20210685]
魏立斐, 刘纪海, 张蕾, 等. 面向隐私保护的集合交集计算综述 [J/OL]. 计算机研究与发展, 2022, 59(8): 1782-1799. [DOI: 10.7544/issn1000-1239.20210685]

[60] RINDAL P, SCHOPPMANN P. VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE[C]. Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II. [S.l.]: Springer, 2021: 901–930. [DOI: 10.1007/978-3-030-77886-6_31].

[61] ORRÙ M, ORSINI E, SCHOLL P. Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection[C]. Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings. [S.l.]: Springer, 2017: 381–396. [DOI: 10.1007/978-3-319-52153-4_22].

[62] PINKAS B, SCHNEIDER T, TKACHENKO O, et al. Efficient Circuit-Based PSI with Linear Communication[C]. Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III. [S.l.]: Springer, 2019: 122–153. [DOI: 10.1007/978-3-030-17659-4_5].

[63] MEADOWS C A. A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party[C]. Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9, 1986. [S.l.]: IEEE Computer Society, 1986: 134–137. [DOI: 10.1109/SP.1986.10022].

[64] FAN L, CAO P, ALMEIDA J M, et al. Summary cache: a scalable wide-area web cache sharing protocol[J]. IEEE/ACM Trans. Netw., 2000, 8(3): 281–293. [DOI: 10.1109/90.851975].

[65] BONOMI F, MITZENMACHER M, PANIGRAHY R, et al. An Improved Construction for Counting Bloom Filters[C]. Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings. [S.l.]: Springer, 2006: 684–695. [DOI: 10.1007/11841036_61].

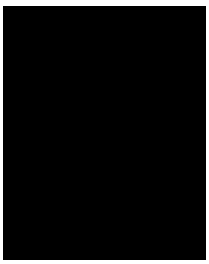
[66] CASACUBERTA S, HESSE J, LEHMANN A. SoK: Oblivious Pseudorandom Functions[C]. European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022. [S.l.] : IEEE, 2022 : 625 –646. [DOI: 10.1109/EuroSP53844.2022.00045].

[67] KILIAN J. Founding Cryptography on Oblivious Transfer[C]. Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. [S.l.] : ACM, 1988 : 20 –31. [DOI: 10.1145/62212.62215].

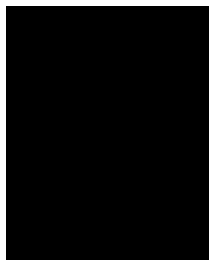
[68] IMPAGLIAZZO R, RUDICH S. Limits on the Provable Consequences of One-Way Permutations[C]. Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA. [S.l.] : ACM, 1989 : 44 –61. [DOI: 10.1145/73007.73012].

[69] RINDAL P, RAGHURAMAN S. Blazing fast PSI from improved OKVS and subfield vole. Cryptology ePrint Archive, Report 2022/320, 2022. https://eprint.iacr.org/2022/320.

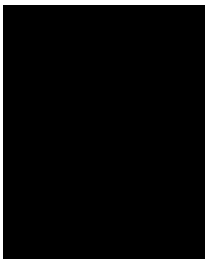
作者信息



作者一 (1989–), 河南郑州人, 博士生在读. 主要研究领域为对称, 密码算法的安全性分析.
zuozhe1@net.cn



作者二 (1982–), 山东济南人, 教授. 主要研究领域为对称, 密码算法的安全性分析.
zuozhe2@net.cn



作者三 (1989–), 北京人, 副研究员. 主要研究领域为对称, 密码算法的安全性分析.
zuozhe3@net.cn