

# 半可信云服务器辅助的高效隐私交集计算协议<sup>\*</sup>

魏立斐<sup>1</sup>, 王勤<sup>1</sup>, 张蕾<sup>1</sup>, 陈聪聪<sup>1</sup>, 陈玉娇<sup>1</sup>, 宁建廷<sup>2,3</sup>



<sup>1</sup>(上海海洋大学 信息学院, 上海 201306)

<sup>2</sup>(福建师范大学 计算机与网络空间安全学院(软件学院), 福建 福州 350117)

<sup>3</sup>(信息安全部国家重点实验室(中国科学院信息工程研究所), 北京 100093)

通信作者: 张蕾, E-mail: [Lzhang@shou.edu.cn](mailto:Lzhang@shou.edu.cn)

**摘要:** 隐私集合交集 (private set intersection, PSI) 是隐私计算中的热点, 其允许参与双方在不泄露任何额外信息的要求下计算交集。现有的隐私集合交集计算方案对参与双方的计算能力要求高, 且计算能力差的参与方无法在保证集合数据隐私的前提下将计算安全外包给云服务器。设计了一种新的不经意两方分布式伪随机函数, 允许半可信的云服务器参与相等性测试, 又不泄露参与方任何集合信息。基于该不经意伪随机函数构建了半可信云服务器辅助的隐私集合交集计算协议, 将主要计算量外包给云服务器。在半诚实模型下证明了协议的安全性。同时, 该协议可保密地计算隐私集合交集的基数。通过与现有协议分析与实验性能比较, 该协议效率高, 计算复杂度与通信复杂度均与集合大小呈线性关系, 适用于客户端设备受限的应用场景。

**关键词:** 隐私集合交集 (PSI); 安全多方计算; 隐私交集基数; 云服务器辅助; 弱客户端

中图法分类号: TP309

中文引用格式: 魏立斐, 王勤, 张蕾, 陈聪聪, 陈玉娇, 宁建廷. 半可信云服务器辅助的高效隐私交集计算协议. 软件学报. <http://www.jos.org.cn/1000-9825/6397.htm>

英文引用格式: Wei LF, Wang Q, Zhang L, Chen CC, Chen YJ, Ning JT. Efficient Private Set Intersection Protocols with Semi-trusted Cloud Server Aided. Ruan Jian Xue Bao/Journal of Software (in Chinese). <http://www.jos.org.cn/1000-9825/6397.htm>

## Efficient Private Set Intersection Protocols with Semi-trusted Cloud Server Aided

WEI Li-Fei<sup>1</sup>, WANG Qin<sup>1</sup>, ZHANG Lei<sup>1</sup>, CHEN Cong-Cong<sup>1</sup>, CHEN Yu-Jiao<sup>1</sup>, NING Jian-Ting<sup>2,3</sup>

<sup>1</sup>(College of Information Technology, Shanghai Ocean University, Shanghai 201306, China)

<sup>2</sup>(College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China)

<sup>3</sup>(State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093, China)

**Abstract:** Private set intersection (PSI) is a hot topic in the privacy-preserving computation, which allows two parties computing the intersection of their sets without revealing any additional information except the resulting intersection. Prior PSI protocols mostly consider the scenario between two parties with the potential limitation of requiring expensive hardware. In addition, the weak client with low computation capability cannot outsource the computation to semi-trusted cloud without keeping the data privacy. This study designs a new oblivious two-party distributed pseudorandom function (Otd-PRF), which allows the semi-trusted cloud servers participating the equality test without any leakage of the set information. Based on Otd-PRF, a cloud-aided PSI protocol is designed which can delegate the major computation to the semi-trusted cloud. A formal security analysis is also provided in the semi-honest model and it is extended to support the computation of the private set intersection cardinality. Through the comparison with the related work, the proposed protocol is superior in the computation and communication complexity. This protocol is linear in the size of the client's set. Its performance analysis shows

\* 基金项目: 国家自然科学基金(61972241, 61802248, 61972094, 62032005); 上海市自然科学基金(18ZR1417300); 上海市高等学校青年骨干教师国内访问学者项目(A1-2007-00-000503); 上海海洋大学骆肇堯大学生科技创新基金(A1-2004-20-201312, A1-2004-21-201311); 福建省科协第二届青年人才托举工程

收稿时间: 2021-01-26; 修改时间: 2021-03-23; 采用时间: 2021-06-08; jos 在线出版时间: 2022-06-15

that the protocol is more friendly to the client with constrained device in the semi-honest model.

**Key words:** private set intersection (PSI); secure multi-party computation; PSI cardinality; cloud server aided; weak client

隐私集合交集 (PSI) 技术是安全多方计算领域的一个重要方面, 允许各自持有私有集合数据的两方在不泄露除交集外任何额外信息的情况下计算出集合交集<sup>[1,2]</sup>. 隐私集合交集基数 (PSI cardinality, PSI-CA) 指计算隐私集合交集元素的数量而不泄露集合元素的信息, 常被应用于对私有分布式数据的分析.PSI 技术作为安全多方计算的基础板块, 在隐私计算业界受到广泛关注. 近几年, PSI 技术发展迅速, 已经应用于许多领域, 如私有通讯录查找<sup>[3,4]</sup>, 广告曝光效果计算<sup>[5]</sup>和新冠接触者追踪<sup>[6]</sup>等.

传统的两方 PSI 协议很高效, 但在一些场景下仍存有潜在的限制, 如当客户持有受限设备 (如移动设备等) 与大型服务提供商之间执行 PSI 协议时, 会出现客户端算力不足或者通信受限等问题. 因此, 将客户端的计算外包给云服务器是一个可行的办法<sup>[7-11]</sup>. 云计算是新一代网络化商业计算模式, 但直接应用云服务器辅助所面临的安全问题也日益凸显. 在传统的两方 PSI 模型中, 互不信任的两方执行 PSI 协议时, 双方会完全信任本地软硬件环境等计算资源. 当客户端将计算委托给半可信云服务器时, 该服务器由外部方运行和管理, 与客户的利益不完全一致, 可能会故意或意外违反数据隐私协定. 因此, 客户无法完全信任第三方云服务器并且将计算委托给半可信云服务器. 理想情况下, 半可信云服务器仅能够辅助计算, 但不应了解有关隐私数据集或计算结果的任何信息. 因此, 设计一个将计算委托给半可信云的 PSI 协议, 不仅必须防止另一方窃取集合数据, 还要保证半可信云服务器不能获取双方集合数据. 现有的方法中往往采用同态加密方案保证外包数据的隐私<sup>[7,9,10]</sup>, 然而当前同态加密算法的性能会降低 PSI 协议的可用性.

本文针对持有受限设备的客户端与大型服务商之间隐私集合交集计算问题, 设计了一种可将隐私相等性测试外包给半可信云的不经意伪随机函数, 并利用 Cuckoo 哈希表、秘密共享、支持异或同态性的不经意传输拓展协议和基于多项式的点集打包方法, 提出了对弱客户端友好的基于半可信云辅助的隐私集合交集协议. 在该协议中, 客户端将自身集合信息经过秘密共享后安全地分发给云服务器和大型服务商, 假定云服务器与大型服务提供商之间不谋, 则在计算过程中云服务器无法获取客户端和服务商的任何敏感信息. 协议运行结束后, 客户端只得到了交集结果, 而服务商对计算结果不得而知.

综上, 本文设计了一个适用于云环境且对弱客户端友好的隐私集合交集协议. 贡献如下:

(1) 为高效地解决隐私集合交集计算安全外包问题, 设计了不经意两方分布式伪随机函数 (oblivious two-party distributed pseudorandom function, Otd-PRF), 使得云服务器能够与服务商保密地对集合元素做相等性测试, 而无需知道客户端集合信息, 也不知道测试结果.

(2) 构造了一个基于半可信云服务器辅助的高效隐私交集计算协议. 该协议适合于弱客户端借助半可信云与服务商保密计算集合交集, 该协议通信复杂度和计算复杂度均为  $O(n)$ , 即与弱客户端集合大小成线性关系, 与服务商集合大小无关, 并且能够高效地计算隐私交集基数.

(3) 搭建了实验仿真环境, 在不同数量级数据输入的情况下分别测试协议各阶段的性能表现. 详细地给出了与当前能高效外包的隐私交集计算协议的性能对比.

## 1 相关工作

### 1.1 传统的 PSI 技术

早期应用系统设计时遇到隐私集合交集问题往往采用的是朴素哈希解决方案, 使用哈希函数对集合元素进行处理, 再求解哈希值集合的交集, 这种解决方案非常的高效, 但极易受到碰撞攻击. 为了解决朴素哈希方案产生的碰撞问题, 需要使用安全的对比方法对比双方元素是否相等, 如果双方各持有  $m$  个元素, 这项工作最坏将需要  $m^2$  次对比. 为减少对比次数, Huang 等人<sup>[12]</sup>提出构建二维哈希表行与行之间执行上述协议, 使得安全对比次数能下降至  $O(n \log n)$ , 即参与双方使用映射域为  $\{0, 1, \dots, n-1\}$  的哈希函数映射各自集合中的每一个元素至有  $n$  行的二维哈希表中, 随后逐行执行 PSI 协议, 使得元素对比次数大幅减少.

Pinkas 等人<sup>[13,14]</sup>和 Kolesnikov 等人<sup>[15]</sup>将其中一个朴素哈希表替换成 Cuckoo 哈希表和存储空间 stash<sup>[16]</sup>, 将 Cuckoo 哈希表中各行的单个元素与朴素哈希表对应行的所有元素进行安全对比, 将大小为  $s$  的 stash 中所有元素与另一方的所有元素做安全对比, 共需要  $O(m) + O(ms)$  次安全对比, 通信复杂度下降为  $\omega(m\lambda)$ . Falk 等人<sup>[17]</sup>将 Pinkas 等人将 Cuckoo 哈希表的方案中对 stash 的操作替换成一种适用于双方集合大小不平衡的专用 PSI 协议<sup>[4]</sup>, 使得通信复杂度从  $\omega(m\lambda)$  下降到  $O(m\lambda)$ . 同期, Pinkas 等人<sup>[18]</sup>使用一种新的 OT Extension 方法同样将通信复杂度下降至  $O(m\lambda)$ . 此外, 陈振华等人<sup>[19]</sup>利用  $(n, n)$  秘密共享将集合交集问题转化为集合相等问题, 使用非加密方法解决隐私交集问题具有较优通信复杂度. 宋祥福等人<sup>[20]</sup>设计了一组支持隐私计算集合交集函数的协议, 能够计算集合交集并集大小以及交集权值和、交集权值方差. Dou 等人把集合问题转化为向量内积问题, 解决了有理数域上各种集合保密计算问题<sup>[21]</sup>. 但上述 PSI 方案对参与双方的计算能力有一定要求, 且计算能力差的参与方无法在保证数据隐私的前提下, 将计算安全地委托给半可信的云服务器.

## 1.2 云辅助的 PSI 技术

随着云计算的兴起, 已有许多基于云辅助的 PSI 协议被提出, 但部分协议已被指出存在一定安全隐患. 2012 年 Kerschbaum 等人<sup>[22]</sup>提出了两种利用单向函数实现抗合谋的外包 PSI 协议, 但是安全性较差, 半可信云服务器能够获得双方集合的交集信息. 2014 年 Liu 等人设计了适用于云辅助的 PSI 协议<sup>[23]</sup>, 但该协议会泄露集合交集基数. 集合交集基数被广泛使用在数据挖掘领域, 泄露交集基数会使得半可信云服务器在不知道交集的情况下仍然能推测很多敏感信息. 因此从隐私的角度来看, 它不应该被泄露. 同年, Kamara 等人提出了云环境下适用于多种敌手模型的 PSI 方案<sup>[24]</sup>, 该方案计算效率非常高, 但会泄露集合交集基数, 解决该问题的改进方案无法将计算委托给半可信云. 2015 年 Zheng 等人<sup>[25]</sup>和 Qiu 等人<sup>[26]</sup>提出的方案中, 客户端也能将计算外包给半可信云服务器, 这两个方法都需要一个可信的第三方初始化参与方的公钥和私钥, 且同样存在上述泄露集合交集基数的问题.

已有一些 PSI 方案能够将计算安全地委托给半可信云, 但委托之后仍然无法解决客户端计算量通信量过大的问题或存在过多的应用限制. Kerschbaum 等人<sup>[7]</sup>为解决文献[22]中存在的安全隐患基于同态加密算法构造了一个 PSI 协议, 计算复杂度与通信复杂度为  $O(n^2)$ . 2016 年李顺东等人<sup>[8]</sup>使用哥德尔编码提出了半诚实模型下保密计算多个集合间交集并集问题的解决方案, 但不能保密计算 2 个集合间的交集, 并且不适用于处理元素域较大的隐私集合问题. 同期, Abadi 等人<sup>[9]</sup>将集合交集求解问题转化为同态加密后的多项式的最大公因式求解问题提出了一个适用于半可信云环境的隐私交集方案, 其允许多个客户端求交集, 将集合外包给半可信云能多次执行 PSI 运算, 由于使用同态加密导致协议计算量较大. 2018 年 Tajima 等人<sup>[10]</sup>基于布隆过滤器和全同态加密技术提出了一个可外包的隐私集合交集基数协议, 该协议使用全同态加密保证数据隐私, 计算代价和通信代价较高. 2019 年 Abadi 等人<sup>[11]</sup>改进原始协议不再使用同态加密技术, 通信与计算复杂度减少到线性水平, 但客户端需做的计算远高于半可信云服务器, 不适用于本地计算能力有限的情况. 上述云环境下的 PSI 协议均无法将客户端的计算安全高效地委托给云服务器, 文献[22–26]已被指出存在安全隐患, 文献[7,9,10]提出的方案使用同态加密技术保证数据安全但导致整体效率较低, 文献[8]无法解决两方客户端外包问题, 文献[11]提出的方案整体高效, 但是客户端有大量的多项式计算, 不适用于本地计算能力受限的情况. 综上所述, 现有适用于云环境的 PSI 方案无法安全高效地解决弱设备客户端的计算外包问题.

## 2 安全模型与定义

### 2.1 系统架构

在基于半可信云服务器辅助的 PSI 协议中, 涉及 3 个参与方: 客户端  $C$ , 服务商  $S$  和云辅助服务器  $H$ , 与文献[9–11]中类似, 我们假定云辅助服务器  $H$  不与服务商  $S$  合谋. 本文方案主要适用于设备受限的客户端  $C$  与有一定计算能力的服务商  $S$  之间计算集合交集, 而将客户端的计算外包给半可信云服务器  $H$  的实际场景. 客户端  $C$  将隐私数据通过秘密共享给云服务器  $H$  和服务商  $S$ , 由云服务器  $H$  和服务商  $S$  完成计算后将交集结果返回给客户端  $C$ . 最终, 客户端  $C$  得到集合交集, 未参与大量复杂计算且通信量少.

## 2.2 安全模型

本文假设所有参与者都为半诚实参与者<sup>[1]</sup>。模拟范例是目前安全多方计算研究中普遍采用的证明方法，该方法将理想安全多方计算协议的安全性与实际安全多方计算协议的安全性进行对比，如果实际协议不比理想协议泄露更多信息则说明实际协议安全<sup>[8]</sup>。参与方  $i$  在执行输入为  $(x, y, z)$  三元组的协议  $\pi$  时，其视图表示为  $VIEW_i^\pi(x, y, z) = (w, r^i, m_1^i, \dots, m_t^i)$ ，其中  $w \in (x, y, z)$  表示参与方  $i$  的输入值， $r_i$  表示参与方  $i$  在执行过程中产生的随机数， $m_i^j$  表示参与方  $i$  收到的第  $j$  个信息。

## 2.3 安全性定义

本文计算协议可表示为  $\pi : \perp \times (\{0, 1\}^*)^N \times (\{0, 1\}^*)^n \rightarrow \perp \times \perp \times f_{\cap}$ ，其中  $\perp$  表示空输入或空输出， $\{0, 1\}^*$  表示输入元素的域， $N$  和  $n$  表示双方集合大小， $f_{\cap}$  表示交集结果。对于每一组输入  $\perp$ 、集合  $X$  和集合  $Y$  分别属于云服务器  $H$ ，客户端  $C$  和服务商  $S$ ，协议执行后云辅助服务器  $H$  输出为空  $\perp$ ，客户端输出交集结果  $f_{\cap}$ ，服务商  $S$  输出为空  $\perp$ 。

**定义 1.** 令  $F$  表示以上描述的确定性函数，协议  $\pi$  保密地计算  $F$  当且仅当存在概率多项式时间模拟器  $Sim_C$ ， $Sim_H$  和  $Sim_S$  在仅有一方输入输出时能够模拟出与实际执行视图不可区分的模拟视图，即：

$$Sim_C(X, f_{\cap}) \stackrel{C}{\equiv} VIEW_C^\pi(X, \perp, Y) \quad (1)$$

$$Sim_H(\perp, \perp) \stackrel{C}{\equiv} VIEW_H^\pi(X, \perp, Y) \quad (2)$$

$$Sim_S(Y, \perp) \stackrel{C}{\equiv} VIEW_S^\pi(X, \perp, Y) \quad (3)$$

其中， $\stackrel{C}{\equiv}$  表示计算不可区分。

## 3 预备知识

本章将对后续用到的符号和用到的概念作简要介绍。 $\kappa$  和  $\lambda$  分别表示计算安全参数和统计安全参数， $[n]$  表示整数集合  $\{1, 2, \dots, n\}$ 。

### 3.1 秘密共享 (secret sharing)

秘密共享指将一个秘密分配给多个参与者，各参与者持有一部分分享值，只有在拥有足够数量分享值时才能解出完整的秘密。本文后续使用了一个基于异或操作的秘密共享方案<sup>[27]</sup>，在该方案中秘密份额由  $n-1$  个与秘密  $s$  长度相同的随机值  $r_1, r_2, \dots, r_{n-1}$  组成，计算  $r_n = r_1 \oplus \dots \oplus r_{n-1} \oplus s$ ， $r_i$  为分享值，收集  $n$  个分享值后可计算  $r_1 \oplus r_2 \oplus \dots \oplus r_n$  恢复秘密  $s$ ，而任何少于  $n$  个分享值的情况都不会泄露秘密  $s$ 。

### 3.2 不经意伪随机函数 (oblivious pseudorandom function, OPRF)

不经意伪随机函数<sup>[28]</sup>是一个安全两方协议，参与对象由接收方和发送方组成。协议执行时，随着接收方输入值  $x$ ，OPRF 协议产生密钥  $k$ 。最终，发送方得到密钥  $k$ ，接收方得到 OPRF 结果值  $F(k, x)$ 。发送方可通过密钥  $k$  计算  $F(k, y)$ ，其中  $y$  为待计算的参数值。OPRF 可被用于测试接收方的私有元素  $x$  是否存在于发送方的私有集合  $Y = \{y_1, y_2, \dots, y_n\}$  中，发送方持有 OPRF 密钥  $k$ ，接收方持有值  $F(k, x)$ ，发送方使用密钥  $k$  计算  $F(k, y_i), i \in [n]$  得到集合  $\{F(k, y_i) | i \in [n]\}$ ，将结果发送给接收方。接收方可通过对是否  $F(k, y_i) = F(k, x)$  从而判断是否存在  $y_i = x$ 。

文献 [15,29] 提出了一种基于少量公钥加密和大量对称密钥加密操作的 OPRF 协议，能以较低的代价产生大量 OPRF 实例。Kolesnikov 等人<sup>[15]</sup>提出了一种 OPRF 密钥为  $(s, k)$  的不经意伪随机函数 BaRK-OPRF。第一个密钥  $s$  是由发送者选择的随机秘密值。第二个密钥的组成结构为  $k = t \oplus [C(x) \wedge s]$ ，其中  $x$  为 OPRF 输入值， $C(\cdot)$  为随机函数， $\wedge$  表示按比特与 (AND)，值  $t$  作为 OPRF 输出由函数随机选择或由接受者选择。最终，接受者能够得到  $F_s(k, x) = t$ 。在 BaRK-OPRF 中，对于每个 OPRF 实例，接收者能够计算唯一的输入值  $x$ ，发送者能够计算任意  $y$  值的 OPRF 结果  $F_s(k, y) = k \oplus [C(y) \wedge s]$ ，将  $k$  值代入后能得到  $F_s(k, y) = t \oplus [(C(y) \oplus C(x)) \wedge s]$ 。如果  $x = y$ ，则  $F_s(k, y) = t$ ，此时， $F_s(k, y) = F_s(k, x)$  成立。

Duong 等人<sup>[6]</sup>以 BaRK-OPRF 为基础提出了满足异或同态性质的不经意分布式密钥伪随机函数 (oblivious

distributed key pseudorandom function, Odk-PRF), 其输入值能够通过秘密共享外包给  $m$  个互不合谋的接收方. Odk-PRF 由一个发送方和  $m$  个接收方组成, 设  $x = x_1 \oplus x_2 \oplus \dots \oplus x_m$ , 接收方  $i$  持有经 XOR 秘密共享后关键字  $x$  的一部分  $x_i$ , 执行 Odk-PRF 协议后, 发送方可得到一个密钥  $s$  和 OPRF 密钥  $k$ ,  $k = k_1 \oplus k_2 \oplus \dots \oplus k_m$ , 接收方  $i$  则得到 OPRF 部分结果  $F_s(k_i, x_i)$ . 当收集  $m$  个 OPRF 部分结果, 可重构出由值  $x$ 、相关密钥  $s$  和密钥  $k$  生成的 OPRF 结果  $F_s(k, x)$ . 其中  $k_i = t_i \oplus [C(x_i) \wedge s]$ , 于是有  $k = k_1 \oplus \dots \oplus k_m = (t_1 \oplus \dots \oplus t_m) \oplus [(C(x_1) \oplus \dots \oplus C(x_m)) \wedge s]$  成立, 采用具有 XOR 同态性的线性编码函数  $C(\cdot)$ <sup>[29]</sup>, 使得  $k = t \oplus [C(x) \wedge s]$  成立. 定义  $F_s(k, x) := t$ , 其中  $t = t_1 \oplus t_2 \oplus \dots \oplus t_m$ . Odk-PRF 协议步骤: (1) 每个客户端  $C_i$  都与服务商  $S$  执行一次 OPRF 函数: 客户端  $C_i$  参与 OPRF 作为接受者输入  $x_i$ ; 服务商  $S$  参与 OPRF 作为发送者得到密钥  $k_i$  和一个在本次函数执行过程中全部相同的相关密钥  $s$ ; 客户端  $C_i$  得到 OPRF 结果值  $t_i$ , 即  $F_s(k_i, x_i)$ ; (2) 服务商计算主密钥  $k = k_1 \oplus k_2 \oplus \dots \oplus k_m$ . 然而, 原始结构的 Odk-PRF 要求  $m$  个客户端不能全部合谋, 因腐败人数达到秘密共享门限便可恢复  $x$  值. Duong 等人<sup>[6]</sup>的工作中假定存在不全合谋的多个云服务器提供计算能力, 但在隐私集合交集计算中往往不需如此庞大的云网络, 其方案若只将计算外包给一台服务器, 却又不能保证客户端集合数据的安全.

### 3.3 布谷鸟哈希 (cuckoo hashing)<sup>[14]</sup>

Cuckoo 哈希算法是一种借助  $k$  个哈希函数  $H_1, H_2, \dots, H_k$  建立密集哈希表的方法, 可用于降低 PSI 协议中的安全对比次数. Cuckoo 哈希算法有一定失败几率, 为防止哈希映射碰撞导致构建失败, 常常以一个特殊的存储空间  $stash$  存放碰撞元素. 文献 [3] 对 Cuckoo 哈希构建的选参进行了详细分析, 实验证明合理选择参数  $\beta, k$  可使得成功构建几率高于  $(1 - 2^{-\lambda})$  而不再需要为防止构建失败生成额外的存储空间  $stash$ .

本文使用 Cuckoo 哈希算法中相同的  $k$  个哈希函数构造朴素哈希表, 将集合  $Y$  映射到长度为  $\beta$  的二维容器中, 其中每个  $y \in Y$  在二维容器中出现  $k$  次. 基于参数  $k, \beta, |Y|$  经随机算法<sup>[3]</sup> 分析能够推测一个上边界  $\eta$ , 使得容器中每行放入元素数量超过上边界  $\eta$  的几率为  $2^{-\lambda}$ .

### 3.4 点集的打包方法

点集信息的打包主要由两个方法组成: (1)  $pack(S) \rightarrow \Pi$ : 集合  $S = \{(a_1, b_1), (a_2, b_2), \dots, (a_\eta, b_\eta)\}$ , 通过该方法处理集合  $S$  后输出一个集合的其他表示形式  $\Pi$ . (2)  $unpack((\Pi, a) \rightarrow v$ : 将  $\Pi$  和关键字  $a$  作为参数, 通过此方法可得到一个输出值  $v$ .

打包方法应该满足以下性质: (1) 正确性: 如果  $(a, b) \in S$  且  $pack(S) \rightarrow \Pi$ , 那么  $(a, unpack(\Pi, a)) \in S$  成立. (2) 不经意性: 当  $b_i$  均匀分布, 按照方法  $pack(\{(a_1, b_1), (a_2, b_2), \dots, (a_\eta, b_\eta)\})$  得到集合的表示形式  $\Pi$ , 对于不相等的  $a_i$  和  $a_j$ ,  $unpack(\Pi, a_i)$  和  $unpack(\Pi, a_j)$  产生的结果应当不可区分.

本文使用文献 [30] 中基于多项式的打包结构和文献 [27] 提出的混淆布隆过滤器作为子模块分别对集合信息进行打包处理, 以多项式打包点集为例,  $pack(S)$  指构建次数为  $(\eta-1)$  的多项式  $\Pi$  来表达  $S = \{(a_1, b_1), (a_2, b_2), \dots, (a_\eta, b_\eta)\}$  点集.  $unpack(\Pi, a)$  方法在此指解出值  $a$  在多项式  $\Pi$  中的对应结果. 以上两种点集打包方法已在文献 [30, 27] 中被证明满足不经意性和正确性. 不同的点集打包方法应用在 PSI 的计算中有不同的性能表现. 若服务商基于多项式打包点集, 则传输  $\Pi$  时仅需发送多项式系数, 通信性能表现较好. 基于混淆布隆过滤器打包的点集结果在传输时表现为一维数组, 相较传输多项式的系数通信稍高, 但其构建速度快, 计算效率表现较好.

## 4 不经意两方分布式伪随机函数

本文将以原始 OPRF 和 Odk-PRF 为基础设计一种不经意两方分布式伪随机函数 (oblivious two-party distributed pseudorandom function, Otd-PRF) 解决本文隐私集合交集计算的外包问题, 其结构如图 1 所示. 在 Otd-PRF 结构中, 随机函数  $C(\cdot)$  均采用已证明安全性的具有 XOR 同态性的线性编码函数<sup>[29]</sup>.

Otd-PRF 协议结构如下:

**协议 1.** 不经意两方分布式伪随机函数 (Otd-PRF)

参数: 服务商  $S$ , 客户端  $C$ , OPRF 函数, 单向哈希函数  $H : \{0, 1\}^* \rightarrow \{0, 1\}^*$ .

输入: 客户端输入  $x_1$ , 服务商输入  $x_2$ .

(1) OPRF 实例生成

客户端  $C$  与服务商  $S$  执行一次 OPRF 函数:

- ① 客户端  $C$  参与 OPRF 作为接受者输入  $x_1$ .
- ② 服务商  $S$  参与 OPRF 作为发送者得到 OPRF 密钥  $k'$  和相关密钥  $s$ , 其中  $k' = t \oplus [C(x_1) \wedge s]$ .
- ③ 客户端  $C$  得到 OPRF 结果值  $t$ , 即  $F(k, x_1 \oplus x_2)$ .



图 1 不经意两方分布式伪随机函数

(2) 服务商  $S$  计算主密钥  $k = k' \oplus [C(x_2) \wedge s]$

① 正确性分析:

$F_s(k, y) = k \oplus [C(y) \wedge s] = t \oplus [C(x_1) \wedge s] \oplus [C(x_2) \wedge s] \oplus [C(y) \wedge s] = t \oplus [(C(x_1 \oplus x_2)) \oplus C(y)) \wedge s] = t \oplus [(C(x) \oplus C(y)) \wedge s]$   
上述  $C(\cdot)$  具有 XOR 同态性, 且  $x_1 \oplus x_2 = x$ . 若  $x = y$ , 则  $F_s(k, y) = t \oplus [0^\kappa \wedge s] = t$ . 若  $x \neq y$ , 则  $C(x) \oplus C(y) \neq 0^\kappa$ , 结果不含有用信息.

② 安全性分析: Otd-PRF 的安全性直接来自其构建模块 OPRF 和秘密共享的安全性. 原始数据  $x$  的安全性在客户端和服务商不合谋的情况下由秘密共享保证. 实际应用中, 接收方常能够得到  $H(k \oplus [C(y) \wedge s])$ , 但由于接收方不知道长度为  $\kappa$  的  $s$ , 且当  $x \neq y$  时,  $C(x)$  和  $C(y)$  汉明距离  $\geq \kappa$ ,  $F_s(k, y)$  对于接收方来说不包含任何隐私信息.

不经意两方分布式伪随机函数 Otd-PRF 与秘密共享配合使用可以解决客户端集合数据的外包计算问题, 客户端将集合元素秘密共享给云服务器和服务商后, 可由云服务器与服务商执行 Otd-PRF 协议而客户端不用再参与计算.

## 5 隐私集合交集计算协议

### 5.1 基本协议

在本方案中, 服务商  $S$  通过点集  $\{(y_1, r_1), (y_2, r_2), \dots, (y_N, r_N)\}$  插值得到唯一的  $(N-1)$  阶多项式  $Poly(\cdot)$ , 其中  $R = \{r_1, r_2, \dots, r_N\}$  是被客户端  $C$  和服务商  $S$  所知的随机数集. 服务商  $S$  将多项式系数发送给云服务器  $H$ , 假设云服务器  $H$  持有元素  $x_i \in X$  并将  $x_i$  代入多项式计算得到  $Poly(x_i) = r'_i$ . 若有  $x_i \in Y$ , 则必有  $r'_i \in R$ . 然而, 由于云服务器对  $R$  一无所知. 因此, 其无法根据  $r'_i$  推测出任何相关信息. 云服务器  $H$  将  $r'_i$  有序发送给客户端  $C$ , 客户端  $C$  将  $r'_i$  与  $R$  中的元素进行比对可知  $x_i$  是否为双方共有. 上述计算方法假定云服务器  $H$  知道  $X$  中的元素, 使用 Otd-PRF 能使云服务器  $H$  计算出多项式结果  $r'_i$  而无需知道集合  $X$ . 客户端  $C$  通过秘密共享将自身元素  $x_{i \in [n]}$  分发给不合谋的半可信云服务器  $H$  和服务商  $S$ . 云服务器  $H$  与服务商  $S$  产生  $n$  个 Otd-PRF 实例. 对于每个 Otd-PRF 实例, 云服务器  $H$  作为 Otd-PRF 的客户端输入  $x_i$  的秘密共享份额得到 OPRF 结果值  $t_i$ , 服务商  $S$  作为 Otd-PRF 的服务商得到 OPRF 密钥对  $(k_{i \in [n]}, s)$ , 后续部分将省略密钥  $s$  的书写, 使用 OPRF 密钥  $k_i$  表示密钥对  $(k_i, s)$ .

**协议 2.** 半可信云辅助的隐私集合交集计算协议 (basic protocol)

参量: 客户端  $C$ , 服务商  $S$ , 云服务器  $H$ ; Otd-PRF 函数.

输入: 客户端  $C$  输入集合  $X = \{x_1, x_2, \dots, x_n\}$ ; 服务商  $S$  输入集合  $Y = \{y_1, y_2, \dots, y_N\}$ ; 云服务器  $H$  无输入.

(1) 初始化

- ① 客户端  $C$  选择随机种子  $r$  和  $seed$  并将其发送给服务商  $S$ .
- ② 客户端  $C$  和服务商  $S$  通过随机种子  $r$  生成相同的数集  $R = \{r_1, r_2, \dots, r_n\} \leftarrow PRG(r)$ .

(2) 数据外包

对于每一个  $x_b \in X$ , 客户端  $C$  通过随机数种子  $seed$  选取随机数  $x_b^2$ , 计算  $x_b^1 = x_b \oplus x_b^2$  并将集合  $X_1 = \{x_1^1, x_2^1, \dots, x_n^1\}$

发送给云服务器  $H$ . 服务商  $S$  通过初始化时收到的  $seed$  计算集合  $X_2 = \{x_1^2, x_2^2, \dots, x_n^2\}$ .

### (3) 服务器计算

① 云服务器  $H$  与服务商  $S$  执行 Otd-PRF 生成大量 Otd-PRF 实例. 对于所有  $b \in [n]$ , 服务商  $S$  得到 Otd-PRF 密钥  $k_b$ , 云服务器  $H$  得到输入  $x_b^1$  的 OPRF 结果值  $t_b$ .

② 服务商  $S$  计算  $y_i \in Y$ ,  $b \in [n]$  的 OPRF 值  $u_{b,i} \leftarrow F(k_b, y_i)$ , 生成点集  $P = \{(H(u_{b,i}), r_b) \mid y_i \in Y, b \in [n]\}$ , 将由点集  $P$  插值生成的多项式  $Poly(\cdot)$  发送给云服务器  $H$ .

③ 云服务器  $H$  收到  $Poly(\cdot)$  后, 对每一个  $b \in [n]$ , 计算  $r'_b \leftarrow Poly(H(t_b))$ , 并将  $R' = \{r'_1, r'_2, \dots, r'_n\}$  发送给客户端  $C$ .

### (4) 客户端得到交集

客户端  $C$  收到  $R' = \{r'_1, r'_2, \dots, r'_n\}$  后, 对每一个  $b \in [n]$ , 若  $r'_b \in R$ , 则  $x_b$  为交集元素.

为避免多项式阶数过高, 本文使用类似于 Pinkas 等人在文献 [14] 中对集合的处理方法将元素映射到二维容器中后再对每行元素分别进行多项式操作. 但集合  $X$  对于半可信云  $H$  和服务商  $S$  属于隐私信息, 客户端需要将集合  $X$  的所有元素逐个映射至长度为  $\beta$  宽度为 1 的容器中, 而服务商将集合  $Y$  中的元素映射到长度为  $\beta$  宽度为  $\eta$  的二维容器中, 客户端  $C$  将容器中的元素秘密地分享给半可信云服务器  $H$  和服务商  $S$ , 之后的多项式操作将对应容器的每行分别进行, 构建的多项式阶数大幅减少, 插值和计算将更加高效.

## 5.2 完整协议

为了优化基本方案, 本文借鉴 Pinkas 等人<sup>[14]</sup>使用的 Cuckoo 哈希和朴素哈希以及点集打包技术<sup>[30,27]</sup>能够减少协议的计算量和通信量. 本节将分 4 个阶段介绍完整的隐私集合交集方案, 模型结构如图 2 所示.

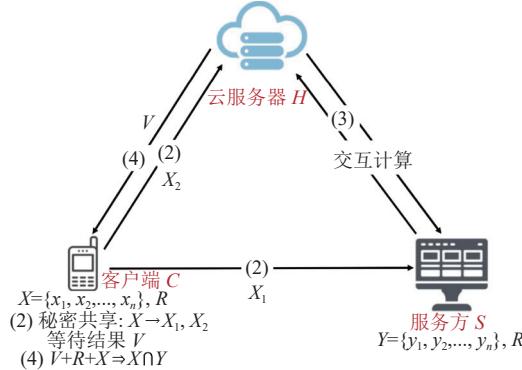


图 2 隐私集合交集计算完整协议

在初始化阶段, 客户端  $C$  选择随机种子  $r$  发送给服务商  $S$ , 双方生成  $\beta$  个随机数  $R = \{r_1, r_2, \dots, r_\beta\} \leftarrow PRG(r)$ . 其中  $\beta$  为构造 Cuckoo 哈希表的长度.

在外包阶段, 客户端  $C$  使用 Cuckoo 哈希算法将集合  $X$  映射到容器  $B_c[\beta]$  中, 随后使用 XOR 秘密共享方案将容器分发给云服务器  $H$  和服务商  $S$ . 云服务器  $H$  得到  $X_1 = \{x_1^1, x_2^1, \dots, x_\beta^1\}$ , 其中  $x_b^1 = x_b \oplus x_b^2$ . 服务商  $S$  则得到一组随机数  $X_2 = \{x_1^2, x_2^2, \dots, x_\beta^2\}$ .  $x_b$  表示  $B_c[b]$  的真实值且  $x_b = x_b^1 \oplus x_b^2$  恒成立.

在服务器计算阶段, 云服务器  $H$  和服务商  $S$  为容器的每行  $b \in [\beta]$  生成一个 Otd-PRF 实例, 服务商  $S$  得到 Otd-PRF 密钥  $k_b$ , 云服务器  $H$  得到  $t_b \leftarrow F(k_b, x_b)$ . 随后服务商  $S$  使用朴素哈希法将集合  $Y$  映射到长度为  $\beta$  宽度为  $\eta$  的二维容器  $B_S[\beta]$  中. 对于容器的每行  $B_S[b]$ ,  $b \in [\beta]$ , 服务商  $S$  使用对应密钥  $k_b$  计算其中  $y_i$  的 OPRF 结果值  $u_{b,i} \leftarrow F(k_b, y_i)$ , 生成一组点集  $P_b = \{(H(u_{b,i}), r_b) \mid y_i \in B_S[b]\}$ , 其中  $H(\cdot)$  为双方共有的单向哈希函数,  $r_b$  为初始化阶段产生的  $R$  集中的随机数. 服务商  $S$  将点集  $P_b$  打包  $\Pi_b \leftarrow pack(P_b)$  并将  $\{\Pi_b \mid b \in [\beta]\}$  发送给云服务器  $H$ , 云服务器  $H$  通过持有的  $t_b$  和收到的  $\{\Pi_b \mid b \in [\beta]\}$  解出  $v_b \leftarrow unpack(\Pi_b, H(t_b))$  并将结果集  $V = \{v_1, v_2, \dots, v_\beta\}$  发送给客户端  $C$ .

对比交集阶段, 客户端  $C$  通过计算  $V \cap R$  可得到 PSI 输出结果  $X \cap Y$ .

### 协议 3. 半可信云服务器辅助的隐私集合交集计算协议 (full protocol)

参量: 客户端  $C$ , 服务商  $S$ , 云服务器  $H$ ; 单项哈希函数  $H: \{0,1\}^* \rightarrow \{0,1\}^*$ ; Cuckoo 哈希算法和朴素哈希算法; Otd-PRF 函数;  $pack()$  和  $unpack()$  方法.

输入: 客户端  $C$  输入集合  $X = \{x_1, x_2, \dots, x_n\}$ ; 服务商  $S$  输入集合  $Y = \{y_1, y_2, \dots, y_N\}$ ; 云服务器  $H$  无输入.

#### (1) 初始化

① 客户端  $C$  选择随机种子  $r$  和  $seed$  并将其发送给服务商  $S$ .

② 客户端  $C$  和服务商  $S$  通过随机种子  $r$  生成相同的数集  $R = \{r_1, r_2, \dots, r_\beta\} \leftarrow PRG(r)$ .

#### (2) 数据外包

① 客户端  $C$  使用 Cuckoo 哈希算法把  $X$  集合映射到长度为  $\beta$  的一维容器中.  $B_c[b]$  表示客户端容器第  $b$  行的元素.

② 对于每一个  $x_b \in B_c[b]$ , 客户端  $C$  通过随机数种子  $seed$  选取随机数  $x_b^2$ , 计算  $x_b^1 = x_b \oplus x_b^2$  并将集合  $X_1 = \{x_1^1, x_2^1, \dots, x_\beta^1\}$  发送给云服务器  $H$ . 服务商  $S$  通过初始化时收到的  $seed$  计算集合  $X_2 = \{x_1^2, x_2^2, \dots, x_\beta^2\}$ .

#### (3) 服务器计算

① 云服务器  $H$  与服务商  $S$  执行 Otd-PRF 生成大量 Otd-PRF 实例. 对于所有  $b \in [\beta]$ , 服务商  $S$  得到 Otd-PRF 密钥  $k_b$ , 云服务器  $H$  得到输入  $x_b^1$  的 OPRF 结果值  $t_b$ .

② 服务商  $S$  使用朴素哈希法将  $Y$  集合映射到长度为  $\beta$  的二维容器中,  $B_S[b]$  表示容器第  $b$  行所有元素组成的集合.

③ 对于所有  $b \in [\beta]$ , 服务商  $S$  计算  $y_i \in B_S[b]$  的 OPRF 值  $u_{b,i} \leftarrow F(k_b, y_i)$ , 生成点集  $P_b = \{(H(u_{b,i}), r_b) \mid y_i \in B_S[b]\}$  后将  $\Pi_b \leftarrow pack(P_b)$  发送给云服务器  $H$ .

④ 云服务器  $H$  收到  $\{\Pi_b \mid b \in [\beta]\}$  后, 计算  $v_b \leftarrow unpack(\Pi_b, H(t_b))$  并将  $V = \{v_1, v_2, \dots, v_\beta\}$  发送给客户端  $C$ .

#### (4) 客户端得到交集

客户端  $C$  收到  $V = \{v_1, v_2, \dots, v_\beta\}$  后, 解出序号集  $O = \{b \mid r_b = v_b, r_b \in R, v_b \in V\}$ , 得到交集  $X \cap Y = \{B_c[b] \mid b \in O\}$ .

本文构造数据交互实例图 3 对协议执行流程做如下说明. 客户端  $C$  将用于生成共有随机数集  $R$  的种子  $r$ , 与用于秘密共享的随机数种子  $seed$  发送给服务商  $S$ , 完成初始化. 随后, 客户端将长度为  $\beta$  的 Cuckoo 哈希表的一个共享  $X_1$  发送给半可信云  $H$ , 而服务商  $S$  可由种子  $seed$  计算出另一个共享  $X_2$ , 完成数据外包. 随后半可信云  $H$  与服务商  $S$  按照 Otd-PRF 协议生成 Otd-PRF 实例, 半可信云  $H$  得到输入  $\{x_1^1, x_2^1, \dots, x_\beta^1\}$  的 OPRF 结果值  $\{t_1, t_2, \dots, t_\beta\}$ , 服务商  $S$  得到 OPRF 密钥  $\{k_1, k_2, \dots, k_\beta\}$  后映射集合  $Y$  构造朴素哈希表, 对于表中每一行  $b$ , 服务商  $S$  计算  $y_i \in B_S[b]$  的 OPRF 值  $u_{b,i} \leftarrow F(k_b, y_i)$ , 生成点集  $P_b = \{(H(u_{b,i}), r_b) \mid y_i \in B_S[b]\}$  计算  $\Pi_b \leftarrow pack(P_b)$ , 将  $\{\Pi_1, \Pi_2, \dots, \Pi_\beta\}$  发送给半可信云  $H$ , 半可信云  $H$  计算  $v_b \leftarrow unpack(\Pi_b, H(t_b))$  将  $V = \{v_1, v_2, \dots, v_\beta\}$  发送给客户端  $C$ , 服务器计算阶段完成. 客户端  $C$  收到  $V = \{v_1, v_2, \dots, v_\beta\}$  解出  $X \cap Y$ , 最终, 客户端  $C$  得到双方隐私集合交集而未参与任何复杂计算.

### 5.3 隐私集合交集基数

计算隐私集合交集基数 (PSI cardinality, PSI-CA) 指保密计算集合交集元素的个数而不泄露集合元素的信息. 客户端  $C$  持有  $X = \{x_1, x_2, \dots, x_n\}$ , 服务商  $S$  持有集合  $Y = \{y_1, y_2, \dots, y_N\}$ , 他们想知道集合交集基数  $|X \cap Y|$  而不想泄露任何集合元素. 广告曝光效果计算<sup>[5]</sup>和接触者追踪<sup>[6]</sup>都有隐私集合交集基数计算的应用场景. 现有支持云外包的交集基数方案效率较低. 在云服务器不与客户端和服务商合谋的情况下, 本文给出的高效隐私交集解决方案经过改造后也可用于保密的求集合交集基数.

隐私交集基数计算方法能以协议 3 为基础改造得到, 即: (1) 初始化中服务商  $S$  生成随机数集后打乱顺序, 生成的随机数集  $\{p_1, p_2, \dots, p_\beta\}$  不再与客户端随机数集  $R = \{r_1, r_2, \dots, r_\beta\}$  顺序一致. (2) 服务器计算中云服务器  $H$  得到的集合  $V$  经过打乱后再发送给客户端  $C$ . 增加的打乱操作使得客户端  $C$  不能得知  $V$  与  $B_c$  位置的对应关系而只能计算  $R$  与  $V$  中有多少元素相同. (3) 客户端  $C$  收到  $V$  后, 解出  $|X \cap Y| = |R \cap V|$ .

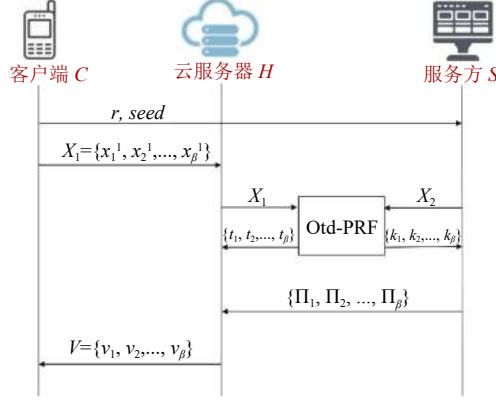


图 3 完整协议中数据交互

#### 5.4 安全性分析

为降低通信量, 本文随机数集均以随机数种子的形式传输, 其安全性与直接传输随机数集一致, 外包数据的安全性由秘密共享保证, 在半可信云和服务商不谋的前提下, 其中任一方均无法窃取客户端的集合数据. 早期方案的碰撞攻击中, 客户端可通过穷举域内所有元素窃取另一方的集合数据. 在本文方案中, 客户端得到的结果集带有随机因子, 其无法在本地通过碰撞攻击得到服务商的隐私数据.

**定理 1.** 基于秘密共享和不经意伪随机函数的安全性, 半诚实模型下本协议安全地实现了借助半可信云保密计算集合交集.

证明: 分别模拟腐败的客户端、腐败的云服务器和腐败服务商以证明本协议在半诚模型下能够保护客户端和服务商的隐私数据. 原始集合  $X$  的安全性与经过 Cuckoo 哈希算法映射后的集合一致, 因此后续使用  $(x_1, x_2, \dots, x_\beta)$  讨论客户端  $C$  集合的安全性.

##### (1) 半诚实客户端

客户端的视图  $VIEW_C^\pi = \{(x_1, x_2, \dots, x_\beta), r, (v_1, v_2, \dots, v_\beta)\}$ , 模拟器  $Sim_C$  扮演云服务器  $H$  和服务商  $S$  按照协议 3 与客户端  $C$  交互,  $Sim_C$  随机选择一个集合  $Y' = \{y'_1, y'_2, \dots, y'_N\}$  并使用朴素哈希法映射到长度为  $\beta$  的二维容器中,  $B_S[b]$  表示容器第  $b$  行中所有元素组成的集合. 随后,  $Sim_C$  生成大量 Otd-PRF 实例. 对于  $b \in [\beta]$ , 得到 Otd-PRF 密钥  $k'_b$  和 OPRF 结果值  $t'_b$ .  $Sim_C$  计算  $y'_i = B_S[b]$  的 OPRF 值  $u'_{b,i} \leftarrow F(k'_b, y'_i)$ , 生成点集  $P'_b = \{(H(u'_{b,i}), r_b) \mid y'_i \in B_S[b]\}$  将点集打包  $\Pi'_b \leftarrow pack(P'_b)$ , 计算  $v'_b \leftarrow unpack(\Pi'_b, H(t'_b))$ , 最终,  $Sim_C$  输出  $((x_1, x_2, \dots, x_\beta), r, (v'_1, v'_2, \dots, v'_\beta))$ .

由于  $Y$  和  $Y'$  都为任意集合, Otd-PRF 具有伪随机性, 因此  $u'_{b,i} \leftarrow F(k'_b, y'_i)$  与  $u_{b,i} \leftarrow F(k_b, y_i)$  不可区分, 点集  $P'_b = \{(H(u'_{b,i}), r_b) \mid y'_i \in B_S[b]\}$  与点集  $P_b = \{(H(u_{b,i}), r_b) \mid y_i \in B_S[b]\}$  不可区分, 打包后的结果  $\Pi'_b$  与  $\Pi_b$  不可区分, 因此  $v'_b \leftarrow unpack(\Pi'_b, H(t'_b))$  与  $v_b \leftarrow unpack(\Pi_b, H(t_b))$  也不可区分, 于是  $(v_1, v_2, \dots, v_\beta)$  与  $(v'_1, v'_2, \dots, v'_\beta)$  不可区分. 因此有  $Sim_C(X, f_{\sqcap}) \stackrel{C}{\equiv} VIEW_C^\pi(X, \perp, Y)$ , 公式 (1) 成立.

##### (2) 半诚实云服务器

云服务器在协议执行全过程收到了由客户端共享的集合  $X_1 = \{x_1^1, x_2^1, \dots, x_\beta^1\}$ 、Otd-PRF 产生的随机值  $t_b$ 、由服务商生成的  $\Pi_b \leftarrow pack(P_b)$  以及经  $unpack(\Pi_b, H(t_b))$  计算得到的结果集  $V = (v_1, v_2, \dots, v_\beta)$ . 考虑以下两种情况:

客户端的安全性: 协议 3 的步骤 (2), 客户将数据通过 XOR 秘密分享后的集合  $X_1 = \{x_1^1, x_2^1, \dots, x_\beta^1\}$  发送给云服务器  $H$ , 假定云服务器  $H$  不与服务商  $S$  合谋, 则基于 XOR 秘密分享的安全性将使得集合  $X_1$  不会泄露客户端集合  $X$  的任何信息, 并且将  $X_1$  替换成任意的  $X_1'$  均不可区分. 由 Otd-PRF 协议的安全性可保证在生成 Otd-PRF 实例时不会泄露除  $t_b$  以外的任何信息给云服务器  $H$ , 实际上  $t_b$  是长度为安全参数  $\kappa$  的随机值, 云服务器  $H$  对  $t_b$  和长度相同的任意随机值  $t'_b$  不可区分.

服务商的安全性: 在协议 3 的步骤 (3) 中, 服务商  $S$  通过点集打包方法  $pack()$  将点集  $P_b = \{(H(u_{b,i}), r_b) \mid y_i \in B_S[b]\}$

打包, 其中  $u_{b,i} \leftarrow F(k_b, y_i)$ ,  $r_b$  为随机数集  $R$  中的随机数. 云服务器  $H$  没有随机数种子  $r$ , 也即没有随机数集合  $R$ , 可将  $r_b$  替换成任意随机值  $r'_b$ , 因此云服务器  $H$  对结果  $\Pi_b$  和  $\Pi'_b$  不可区分.

综上, 云服务器  $H$  视图  $VIEW_H^x(X, \perp, Y) = \{\perp, (x_1^1, x_2^1, \dots, x_\beta^1), (t_1, t_2, \dots, t_\beta), (\Pi_1, \Pi_2, \dots, \Pi_\beta)\}$  与  $Sim_H(\perp, \perp)$  不可区分, 满足公式(2).

### (3) 半诚实服务商

同理, 基于 Odt-PRF 的安全性和秘密共享的安全性, 用类似的方法可构造模拟器  $Sim_S$ , 使得公式(3)成立.

综上, 协议3在半诚实模型下借助半可信云保密计算了隐私集合交集. 隐私交集基数计算方法安全性同理可证.

## 6 协议性能与比较

### 6.1 实验环境与参数

本文实验环境为: Ubuntu 18.04.4 LTS, Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz, 16 GB 内存. 本文使用 C++语言实现了基于半可信云服务器辅助的隐私交集计算协议, 并与 Abadi 等人 EO-PSI 方案<sup>[11]</sup>进行对比. 为了提升计算效率, 本文在对基于多项式的打包结构进行实验时使用了 NTL 库<sup>[31]</sup>, 实现 Otd-PRF 时借助了 libOTe<sup>[32]</sup>库提供的 1-out-of-N 不经意传输拓展协议.

具体实现时, 集合元素长为 128 比特, 统计安全参数  $\lambda = 40$ , 计算安全参数  $\kappa = 128$ , 与现有大多数 PSI 方案取值相同. Cuckoo 哈希算法参数选择: 由文献[3]对 Cuckoo 哈希算法的参数取值分析可知  $\beta = 1.5n$ ,  $k = 3$  时可使得成功构建几率大于  $(1 - 1^{-\lambda})$  且不再需要为防止构建失败生成额外的存储空间 stash, 本文方案实现时客户端将使用 3 个哈希函数将集合元素映射到大小为  $\beta = 1.5n$  的 Cuckoo 哈希表中, 在统计上能够保证 Cuckoo 哈希表成功构建, 同时, 服务商使用相同的 3 个哈希函数通过朴素哈希法映射集合元素. 混淆布隆过滤器参数选择: 文献[27]和文献[6]对布隆过滤器的假阳性问题做了详细的分析, 证明了当选用 31 个哈希函数构建大小为  $58N$  的混淆布隆过滤器时, 能保证假阳性出现的概率接近  $2^{-\lambda}$ , 故本文选择同样的参数.

### 6.2 性能评估与分析

本文分别基于两种点集打包结构对协议进行完整实现, 统计协议各阶段计算时间和通信代价, 结果如表1所示, 客户端需执行的数据外包和交集结果解出步骤, 在总通信代价和计算代价中占比低, 使用多项式打包结构(Poly.)的隐私交集协议通信量少, 而使用混淆布隆过滤器(GBF)的协议运算速度快.

表1 完整协议的分阶段性能评测

参数	运行时间 (s)						通信量 (MB)															
	集合大小	$N$			$2^{16}$			$2^{18}$			$2^{20}$			$2^{16}$			$2^{18}$			$2^{20}$		
		$n$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$	$2^{12}$	$2^{16}$				
数据外包		0.0003	0.0053	0.0002	0.0057	0.0003	0.0052	0.09	1.5	0.09	1.5	0.09	1.5	0.09	1.5	0.09	1.5	0.09	1.5			
Otd-PRF		0.1	0.4	0.2	0.7	0.8	1.6	0.38	6.01	0.38	6.01	0.38	6.01	0.38	6.01	0.38	6.01	0.38	6.01			
Pack与Unpack	Poly.	5.5	29.7	20.8	39.0	76.8	98.6	<b>0.05</b>	<b>0.75</b>													
	GBF	<b>0.4</b>	<b>0.4</b>	<b>1.2</b>	<b>1.4</b>	<b>4.5</b>	<b>4.5</b>	58	58	232	232	928	928	928	928	928	928	928	928			
交集结果解出		0.0005	0.007	0.0005	0.007	0.0005	0.007	0.09	1.5	0.09	1.5	0.09	1.5	0.09	1.5	0.09	1.5	0.09	1.5			
总和	Poly.	5.6	30.2	21.1	39.7	77.7	100.3	<b>0.61</b>	<b>9.76</b>													
	GBF	<b>0.5</b>	<b>0.9</b>	<b>1.4</b>	<b>2.1</b>	<b>5.3</b>	<b>6.1</b>	58.56	67.01	232.56	241.01	928.56	937.01									

### 6.3 性能对比

计算复杂度和通信复杂度是隐私集合交集计算协议的重要评价指标, 借助半可信云服务完成隐私集合交集计算还需要考虑是否能防止云服务器对私有数据的好奇、能否将集合数据安全的存储在半可信云服务器上、能否将计算外包给云服务器这 3 个重要评价指标, 对于一个云环境下的 PSI 协议, 若云服务器在半诚实模型下无法窃取任何信息则说明能够防止云服务器对私有数据的好奇, 数据经过处理后能供云服务器多次使用则表示支持数据

存储外包, 若外包之后本地计算量小于云服务器则说明协议支持计算外包。表 2 列出了本文方案与现有云环境下隐私集合交集协议的综合对比。表中  $c$  表示集合大小。

表 2 云环境下隐私集合交集计算协议对比

协议	抗半可信云窃取	数据外包	计算外包	通信复杂度	计算复杂度
Kerschbaum等人 <sup>[22]</sup>	否	否	是	$O(c^2)$	$O(c^2)$
Kerschbaum等人 <sup>[7]</sup>	是	否	是	$O(c^2)$	$O(c^2)$
Liu等人 <sup>[23]</sup>	否	是	是	$O(c)$	$O(c^2)$
Kamara等人 <sup>[24]</sup>	否	否	否	$O(c)$	$O(c)$
Zheng等人 <sup>[25]</sup>	否	是	是	$O(c)$	$O(c)$
Qiu等人 <sup>[26]</sup>	否	是	否	$O(c)$	$O(c)$
Abadi等人 <sup>[9]</sup>	是	是	否	$O(c)$	$O(c^2)$
Tajima等人 <sup>[10]</sup>	是	否	否	$O(c^2)$	$O(c^2)$
Abadi等人 <sup>[11]</sup>	是	是	否	$O(c)$	$O(c)$
本文方案	是	否	是	$O(c)$	$O(c)$

文献 [22–26] 中提出的云环境下的隐私集合交集方案存在一定安全隐患, 会向半可信云服务器泄露部分敏感信息。现有工作仅有文献 [7] 提出的方案支持安全的计算外包, 但由于使用同态加密保证数据安全, 该方案有较大计算量, 方案整体效率低。文献 [9,11,26] 中提出的方案能够将数据安全存储在半可信云服务器上, 但客户端将数据外包至云服务器后仍有超过外包数据的通信量, 且双方客户端在计算时需要保持在线。Abadi 等人提出的 EO-PSI<sup>[11]</sup>能够保证参与方集合数据的隐私, 由于没有使用同态加密技术, 该方案相比之前方案高效。本文以 EO-PSI 为对比性能有较大提高, 各阶段的运行时间和通信量如表 3 所示。

表 3 运行时间与通信量对比

阶段	协议	运行时间 (s)					通信量 (MB)				
		$2^{12}$	$2^{14}$	$2^{16}$	$2^{18}$	$2^{20}$	$2^{12}$	$2^{14}$	$2^{16}$	$2^{18}$	$2^{20}$
数据外包	EO-PSI <sup>[11]</sup>	0.2	0.7	1.9	11.9	47.2	0.34	1.34	5.45	22.34	89.35
	本文方案	<b>0.0002</b>	<b>0.001</b>	<b>0.005</b>	<b>0.03</b>	<b>0.3</b>	<b>0.09</b>	<b>0.38</b>	<b>1.5</b>	<b>6</b>	<b>24</b>
云辅助计算	EO-PSI <sup>[11]</sup>	0.3	1.2	3.3	20.8	83.6	<b>0.34</b>	<b>1.34</b>	<b>5.45</b>	<b>22.34</b>	<b>89.35</b>
	本文方案	Poly. <b>0.063</b>	6.3 <b>0.3</b>	29.3 <b>0.9</b>	121.2 <b>3.6</b>	484.3 <b>14.6</b>	0.43	1.69	6.76	27.05	108.18
交集结果解出	EO-PSI <sup>[11]</sup>	9.1	36.3	103.5	604.5	2460.6	0.34	1.34	5.45	22.34	89.35
	本文方案	<b>0.0005</b>	<b>0.002</b>	<b>0.007</b>	<b>0.04</b>	<b>0.1</b>	<b>0.09</b>	<b>0.38</b>	<b>1.5</b>	<b>6</b>	<b>24</b>
总和	EO-PSI <sup>[11]</sup>	9.7	38.3	108.8	637.3	2591.5	1.02	4.02	16.35	67.02	268.05
	本文方案	Poly. <b>0.064</b>	6.3 <b>0.3</b>	29.3 <b>0.9</b>	121.2 <b>3.6</b>	484.7 <b>15</b>	<b>0.61</b>	<b>2.45</b>	<b>9.76</b>	<b>39.05</b>	<b>156.18</b>

## 7 结 论

隐私集合的交集计算是隐私计算中的热点, 具有广泛的实际应用场景。本文设计了一种不经意两方分布式伪随机函数 (Otd-PRF), 使半可信云服务器参与相等性测试又无需知道集合信息。根据 Otd-PRF 设计了将计算外包给半可信云服务器的隐私集合交集计算协议, 当服务商和云服务器不谋时能够高效地计算服务商与客户端之间的隐私集合交集和交集基数, 尤其适用于一方计算能力弱但能够租用半可信云服务器辅助计算的实际场景。在未来的工作中, 我们将进一步研究将数据外包给半可信云服务器, 实现一次外包多次使用, 能够减少客户端的通信代价。

### References:

- [1] Shen LY, Chen XJ, Shi JQ, Hu LL. Survey on private preserving set intersection technology. Journal of Computer Research and

- Development, 2017, 54(10): 2153–2169 (in Chinese with English abstract). [doi: [10.7544/issn1000-1239.2017.20170461](https://doi.org/10.7544/issn1000-1239.2017.20170461)]
- [2] Cui HR, Liu TY, Yu Y. A survey on private set intersection. Information Security and Communications Privacy, 2019, (3): 48–67 (in Chinese with English abstract). [doi: [10.3969/j.issn.1009-8054.2019.03.010](https://doi.org/10.3969/j.issn.1009-8054.2019.03.010)]
- [3] Demmler D, Rindal P, Rosulek M, Trieu N. PIR-PSI: Scaling private contact discovery. Proc. on Privacy Enhancing Technologies, 2018, 2018(4): 159–178. [doi: [10.1515/popets-2018-0037](https://doi.org/10.1515/popets-2018-0037)]
- [4] Chen H, Laine K, Rindal P. Fast private set intersection from homomorphic encryption. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 1243–1255. [doi: [10.1145/3133956.3134061](https://doi.org/10.1145/3133956.3134061)]
- [5] Lv SY, Ye JH, Yin SJ, Cheng XC, Feng C, Liu XY, Li R, Li ZH, Liu ZL, Zhou L. Unbalanced private set intersection cardinality protocol with low communication cost. Future Generation Computer Systems, 2020, 102: 1054–1061. [doi: [10.1016/j.future.2019.09.022](https://doi.org/10.1016/j.future.2019.09.022)]
- [6] Duong T, Phan DH, Trieu N. Catalic: Delegated PSI cardinality with applications to contact tracing. In: Proc. of the 26th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Daejeon: Springer, 2020. 870–899. [doi: [10.1007/978-3-030-64840-4\\_29](https://doi.org/10.1007/978-3-030-64840-4_29)]
- [7] Kerschbaum F. Outsourced private set intersection using homomorphic encryption. In: Proc. of the 7th ACM Symp. on Information, Computer and Communications Security. Seoul: ACM, 2012. 85–86. [doi: [10.1145/2414456.2414506](https://doi.org/10.1145/2414456.2414506)]
- [8] Li SD, Zhou SF, Guo YM, Dou JW, Wang DS. Secure set computing in cloud environment. Ruan Jian Xue Bao/Journal of Software, 2016, 27(6): 1549–1565 (in Chinese with English abstract). <http://www.jos.org.cn/1000-9825/4996.htm> [doi: [10.13328/j.cnki.jos.004996](https://doi.org/10.13328/j.cnki.jos.004996)]
- [9] Abadi A, Terzis S, Dong CY. O-PSI: Delegated private set intersection on outsourced datasets. In: Proc. of the 30th IFIP Int'l Information Security and Privacy Conf. Hamburg: Springer, 2015. 3–17. [doi: [10.1007/978-3-319-18467-8\\_1](https://doi.org/10.1007/978-3-319-18467-8_1)]
- [10] Tajima A, Sato H, Yamana H. Outsourced private set intersection cardinality with fully homomorphic encryption. In: Proc. of the 6th Int'l Conf. on Multimedia Computing and Systems (ICMCS). Rabat: IEEE, 2018. 1–8. [doi: [10.1109/ICMCS.2018.8525881](https://doi.org/10.1109/ICMCS.2018.8525881)]
- [11] Abadi A, Terzis S, Metere R, Dong CY. Efficient delegated private set intersection on outsourced private datasets. IEEE Trans. on Dependable and Secure Computing, 2019, 16(4): 608–624. [doi: [10.1109/TDSC.2017.2708710](https://doi.org/10.1109/TDSC.2017.2708710)]
- [12] Huang Y, Evans D, Katz J. Private set intersection: Are garbled circuits better than custom protocols? In: Proc. of the 19th Network and Distributed Security Symp. San Diego: ISOC, 2012.
- [13] Pinkas B, Schneider T, Zohner M. Scalable private set intersection based on OT extension. ACM Trans. on Privacy and Security, 2018, 21(2): 7. [doi: [10.1145/3154794](https://doi.org/10.1145/3154794)]
- [14] Pinkas B, Schneider T, Segev G, Zohner M. Phasing: Private set intersection using permutation-based hashing. In: Proc. of the 24th USENIX Conf. on Security Symp. Berkeley: USENIX Association, 2015. 515–530.
- [15] Kolesnikov V, Kumaresan R, Rosulek M, Trieu N. Efficient batched oblivious PRF with applications to private set intersection. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 818–829. [doi: [10.1145/2976749.2978381](https://doi.org/10.1145/2976749.2978381)]
- [16] Kirsch A, Mitzenmacher M, Wieder U. More robust hashing: Cuckoo hashing with a stash. SIAM Journal on Computing, 2010, 39(4): 1543–1561. [doi: [10.1137/080728743](https://doi.org/10.1137/080728743)]
- [17] Falk BH, Noble D, Ostrovsky R. Private set intersection with linear communication from general assumptions. In: Proc. of the 18th ACM Workshop on Privacy in the Electronic Society. London: ACM, 2019. 14–25. [doi: [10.1145/3338498.3358645](https://doi.org/10.1145/3338498.3358645)]
- [18] Pinkas B, Rosulek M, Trieu N, Yanai A. SpOT-light: Lightweight private set intersection from sparse OT extension. In: Proc. of the 39th Annual Int'l Cryptology Conf. Santa Barbara: Springer, 2019. 401–431. [doi: [10.1007/978-3-03-26954-8\\_13](https://doi.org/10.1007/978-3-03-26954-8_13)]
- [19] Chen ZH, Li SD, Huang Q, Ding Y, Liu YR. Secure computation of two set-relationships with the unencrypted method. Ruan Jian Xue Bao/Journal of Software, 2018, 29(2): 473–482 (in Chinese with English abstract). <http://www.jos.org.cn/1000-9825/5262.htm> [doi: [10.13328/j.cnki.jos.005262](https://doi.org/10.13328/j.cnki.jos.005262)]
- [20] Song XF, Gai M, Zhao SN, Jiang H. Privacy-preserving statistics protocol for set-based computation. Journal of Computer Research and Development, 2020, 57(10): 2221–2231 (in Chinese with English abstract). [doi: [10.7544/issn1000-1239.2020.20200444](https://doi.org/10.7544/issn1000-1239.2020.20200444)]
- [21] Dou JW, Liu XH, Wang WL. Privacy preserving two-party rational set computation. Chinese Journal of Computers, 2020, 43(8): 1397–1413 (in Chinese with English abstract). [doi: [10.11897/SP.J.1016.2020.01397](https://doi.org/10.11897/SP.J.1016.2020.01397)]
- [22] Kerschbaum F. Collusion-resistant outsourcing of private set intersection. In: Proc. of the 27th Annual ACM Symp. on Applied Computing. Trento: ACM, 2012. 1451–1456. [doi: [10.1145/2245276.2232008](https://doi.org/10.1145/2245276.2232008)]
- [23] Liu F, Ng WK, Zhang W, Giang DH, Han SG. Encrypted set intersection protocol for outsourced datasets. In: Proc. of the 2014 IEEE Int'l Conf. on Cloud Engineering. Boston: IEEE, 2014. 135–140. [doi: [10.1109/IC2E.2014.18](https://doi.org/10.1109/IC2E.2014.18)] [doi: [10.1109/IC2E.2014.18](https://doi.org/10.1109/IC2E.2014.18)]
- [24] Kamara S, Mohassel P, Raykova M, Sadeghian S. Scaling private set intersection to billion-element sets. In: Proc. of the 18th Int'l Conf. on Financial Cryptography and Data Security. Christ Church: Springer, 2014. 195–215. [doi: [10.1007/978-3-662-45472-5\\_13](https://doi.org/10.1007/978-3-662-45472-5_13)]

- [25] Zheng QJ, Xu SH. Verifiable delegated set intersection operations on outsourced encrypted data. In: Proc. of the 2015 IEEE Int'l Conf. on Cloud Engineering. Tempe: IEEE, 2015. 175–184. [doi: [10.1109/IC2E.2015.38](https://doi.org/10.1109/IC2E.2015.38)]
- [26] Qiu S, Liu JQ, Shi YF, Li M, Wang W. Identity-based private matching over outsourced encrypted datasets. IEEE Trans. on cloud Computing, 2018, 6(3): 747–759. [doi: [10.1109/TCC.2015.2511723](https://doi.org/10.1109/TCC.2015.2511723)]
- [27] Dong CY, Chen LQ, Wen ZK. When private set intersection meets big data: An efficient and scalable protocol. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. Berlin: ACM, 2013. 789–800. [doi: [10.1145/2508859.2516701](https://doi.org/10.1145/2508859.2516701)]
- [28] Freedman MJ, Ishai Y, Pinkas B, Reingold O. Keyword search and oblivious pseudorandom functions. In: Proc. of the 2nd Theory of Cryptography Conf. Cambridge: Springer, 2005. 303–324. [doi: [10.1007/978-3-540-30576-7\\_17](https://doi.org/10.1007/978-3-540-30576-7_17)]
- [29] Orrù M, Orsini E, Scholl P. Actively secure 1-out-of-N OT extension with application to private set intersection. In: Proc. of the Cryptographers' Track at the RSA Conf. San Francisco: Springer, 2017. 381–396. [doi: [10.1007/978-3-319-52153-4\\_22](https://doi.org/10.1007/978-3-319-52153-4_22)]
- [30] Kolesnikov V, Matanis N, Pinkas B, Rosulek M, Trieu N. Practical multi-party private set intersection from symmetric-key techniques. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 1257–1272. [doi: [10.1145/3133956.3134065](https://doi.org/10.1145/3133956.3134065)]
- [31] Shoup V. NTL: A library for doing number theory. <http://www.shoup.net/ntl/>
- [32] Rindal P. libOTe: An efficient, portable, and easy to use Oblivious Transfer Library. <https://github.com/osu-crypto/libOTe>

#### 附中文参考文献:

- [1] 申立艳, 陈小军, 时金桥, 胡兰兰. 隐私保护集合交集计算技术研究综述. 计算机研究与发展, 2017, 54(10): 2153–2169. [doi: [10.7544/issn1000-1239.2017.20170461](https://doi.org/10.7544/issn1000-1239.2017.20170461)]
- [2] 崔泓睿, 刘天怡, 郁昱. 带隐私保护的集合交集计算协议的发展现状综述. 信息安全与通信保密, 2019, (3): 48–67. [doi: [10.3969/j.issn.1009-8054.2019.03.010](https://doi.org/10.3969/j.issn.1009-8054.2019.03.010)]
- [8] 李顺东, 周素芳, 郭奕曼, 窦家维, 王道顺. 云环境下集合隐私计算. 软件学报, 2016, 27(6): 1549–1565. <http://www.jos.org.cn/1000-9825/4996.htm> [doi: [10.13328/j.cnki.jos.004996](https://doi.org/10.13328/j.cnki.jos.004996)]
- [19] 陈振华, 李顺东, 黄琼, 丁勇, 刘娅茹. 非加密方法安全计算两种集合关系. 软件学报, 2018, 29(2): 473–482. <http://www.jos.org.cn/1000-9825/5262.htm> [doi: [10.13328/j.cnki.jos.005262](https://doi.org/10.13328/j.cnki.jos.005262)]
- [20] 宋祥福, 盖敏, 赵圣楠, 蒋瀚. 面向集合计算的隐私保护统计协议. 计算机研究与发展, 2020, 57(10): 2221–2231. [doi: [10.7544/issn1000-1239.2020.20200444](https://doi.org/10.7544/issn1000-1239.2020.20200444)]
- [21] 窦家维, 刘旭红, 王文丽. 有理数域上两方集合的高效保密计算. 计算机学报, 2020, 43(8): 1397–1413. [doi: [10.11897/SP.J.1016.2020.01397](https://doi.org/10.11897/SP.J.1016.2020.01397)]



魏立斐(1982—), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为隐私保护, 应用密码学.



陈聪聪(1996—), 男, 硕士, CCF 学生会员, 主要研究领域为隐私保护, 安全多方计算, 机器学习安全.



王勤(1996—), 男, 硕士, 主要研究领域为安全多方计算, 隐私计算, 信息安全.



陈玉娇(1996—), 女, 硕士, CCF 学生会员, 主要研究领域为隐私保护, 安全多方计算, 机器学习安全.



张雷(1983—), 女, 博士, 讲师, 主要研究领域为信息安全, 访问控制.



宁建廷(1988—), 男, 博士, 教授, 主要研究领域为密码学与数据安全.