中国科学 : 信息科学 2026 年 第 56 卷 第 2 期: 362–377
SCIENTIA SINICA Informationis
论文

基于不经意多项式评估的隐私集合求交方案

亢欣[1], 曾勇[1*], 王珠珠[2*], 马卓然[1], 刘谭仁[1], 周俊杰[1], 刘志宏[1], 马卓[1], 马建峰[1]

1. 西安电子科技大学网络与信息安全学院, 西安 710071
2. 西北大学信息科学与技术学院, 西安 710068
* 通信作者. E-mail: yzeng@mail.xidian.edu.cn, zzwang@nwu.edu.cn

收稿日期: 2025–05–13; 修回日期: 2025–06–30; 接受日期: 2025–08–28; 网络出版日期: 2026–01–26

国家重点研发计划 (批准号: 2023YFE0111100)、国家自然科学基金 (批准号: U21A20464, U23A20306, U23A20307, 62261160651, U2436206, 62406239, 62436002)、中国博士后科学基金 (批准号: 2023M742739)、高等学校学科创新引智计划 (批准号: B16037) 和中央高校基本科研业务费专项资金 (批准号: YJSJ25011, QTZX24081) 资助项目

摘要 传统隐私集合求交方案使得参与方在不泄漏各自隐私集合的情况下计算各自集合的交集, 而无法支持基于交集数据的拓展计算. 电路隐私集合求交方案使得参与方在不泄漏交集数据的前提下, 能够完成基于交集数据的对称函数计算, 从而构建通用的隐私集合求交方案. 首先通过对现有方案进行分析和总结, 提出电路隐私集合求交的计算框架, 其由三部分组成: 数据集本地分类、分类数据不经意编码及交集成员检测与共享, 其中后两部分占总计算开销的 90% 以上, 占总通信开销的 100%. 因此, 本文提出了基于不经意多项式评估实现分类数据的不经意编码. 通过将开销较大的插值多项式计算替换为高效的一阶多项式评估, 将不经意编码的计算复杂度从 $\mathcal{O}(n^2)$ 降低到 $\mathcal{O}(n)$. 此外, 本文设计了一种通信高效的批量相等检测协议, 用于交集成员的快速检测与共享. 通过构造不经意索引偏移方案, 将批量相等检测问题转换为使用单一索引的不经意检索问题, 实现了目前最优的在线通信复杂度. 实验结果表明, 通过以上优化, 本文提出的隐私集合求交方案在运行时间方面相比于 PSTY19 (EUROCRYPT 2019), RS21 (EUROCRYPT 2021) 和 CGS22 (PETs 2022) 分别实现了 2.3, 1.16 和 1.37 倍的加速. 在通信开销方面, 本文所提协议的通信开销仅是 PSTY19 的 31%, RS21 的 29% 和 CGS22 的 52%.

关键词 隐私集合求交, 电路隐私集合求交, 不经意多项式评估, 批量相等检测

1 引言

隐私集合求交 (private set intersection, PSI) [1~4] 使得两个参与方在不泄漏非交集元素的前提下计算各自隐私集合的交集. 作为安全多方计算的关键组件之一, 其被广泛应用于私有联系人发现 [5]、联合营销 [6~8] 和隐私保护机器学习 [9~11] 等多个场景. 然而, 标准 PSI 方案直接向参与方输出两个数据集的交集, 而无法支持更灵活的基于交集数据的拓展隐私计算, 如计算交集个数 [12]、求和 [13] 及基于阈值的计算 [14,15]. 为了突破这一局限性, 电路隐私集合求交方案 (circuit-based PSI, CPSI) [16,17] 被提出, 其允许参与方在不泄漏交集数据的前提下, 完成

引用格式: 亢欣, 曾勇, 王珠珠, 等. 基于不经意多项式评估的隐私集合求交方案. 中国科学: 信息科学, 2026, 56: 362–377, doi: 10.1360/SSI-2025-0184
Kang X, Zeng Y, Wang Z Z, et al. Privacy set intersection from oblivious polynomial evaluation. Sci Sin Inform, 2026, 56: 362–377, doi: 10.1360/SSI-2025-0184

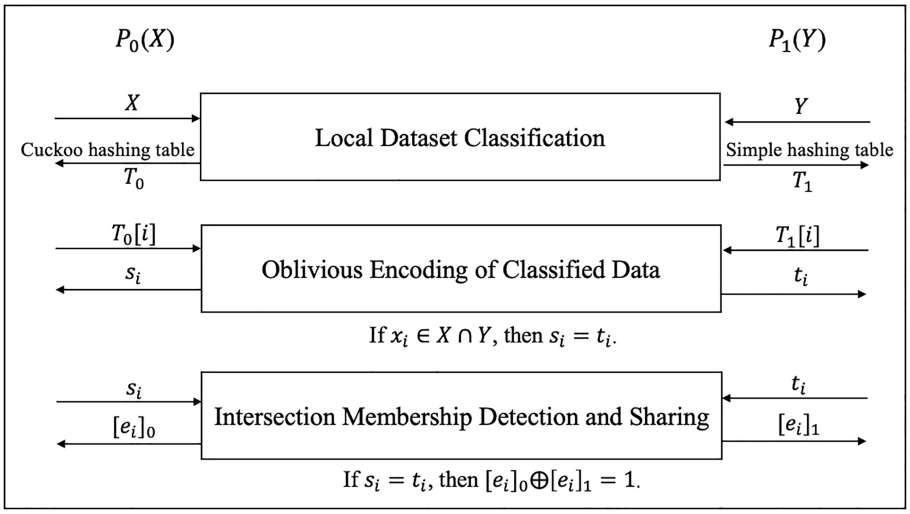

图 1　电路隐私集合求交协议框架图.

Figure 1　The framework of circuit-based PSI.

基于集合交集数据的对称函数计算, 从而实现通用的 PSI 方案. 由于其更强的安全性和拓展的计算能力, CPSI 被应用于协同数据分析 [18, 19]、疾病接触检测 [20, 21] 和冗余数据删除 [22] 等重要场景.

为了进一步优化和提升 CPSI 的效率, 本文首先对现有方案 [16, 17, 23, 24] 进行系统的总结与归纳, 并提出 CPSI 的计算框架, 如图 1 所示. 具体而言, 本文将现有方案的实现划分为以下 3 个核心组件: 数据集本地分类、分类数据不经意编码及交集成员检测与共享.

• 数据集本地分类. P_0 和 P_1 分别采用不同的映射算法对本地数据集元素进行分类与存储, 确保交集元素在不同的存储结构中被置于相近的位置.

• 分类数据不经意编码. 基于上述存储位置, P_0 和 P_1 使用不经意编码机制对不同存储结构的每个位置的数据进行编码. 对于交集元素, 两个参与方获得相同的随机值; 否则, 获得两个不相关的随机值.

• 交集成员检测与共享. 在此阶段, P_0 与 P_1 根据所接收到的随机值判断集合数据是否属于交集. 若该数据属于交集, 双方将获得 1 的布尔共享值; 否则, 获得 0 的布尔共享值.

基于图 1 所示的 CPSI 框架, 本文通过实验与理论分析得出如下结论. 第一部分由各参与方本地完成, 因此协议的总通信开销完全由第二、三部分决定. 此外, 实验结果表明, 这两个部分的计算开销占协议总计算开销的 90% 以上. 因此, 优化第二、三部分的性能对于提升 CPSI 的整体效率具有重要意义.

分类数据的不经意编码. 现有分类数据的不经意编码主要基于多项式插值 (poly-nomials interpolation, PNI) 与不经意键值存储 (oblivious key-value store, OKVS) 两种原语实现. 其中, 基于 PNI 的实现需要求解 n 个线性多项式方程, 其计算复杂度为 $\mathcal{O}(n^2)$. 此外, 基于 OKVS 的实现通过构造稀疏矩阵进行效率优化, 将计算复杂度降至 $\mathcal{O}(n \log n)$. 但在数据编码过程中, OKVS 通常需构建额外的辅助扩展向量, 此部分将带来约 20%∼40% 的存储开销.

因此, 本文提出了一种基于不经意多项式评估 (oblivious polynomial evaluation, OPE) 的新型编码方式. 该方法通过对乘积多项式进行高效的不经意评估, 在不引入额外存储开销的前提下, 将计算复杂度降低至 $\mathcal{O}(n)$. 具体地, P_1 根据各存储位置的数据构造乘积多项式, 使 P_0 能够基于其隐私数据对该多项式进行评估, 而不暴露具体的多项式构造. 对于交集数据, P_0 的多项式评估结果将与 P_1 在构造多项式时引入的随机值一致; 否则, 所得结果为相互独立且均匀分布的随机数. OPE 是实现高阶多项式不经意评估的基础原语. 然而, 现有 PSI 协议中的 OPE 方案主要基于同态加密实现, 其需要在密文数据上进行复杂的高阶计算, 导致昂贵的计算开销. 为了优化 OPE 在高阶多项式下的计算效率, 本文提出了一种多项式递归分解方法, 使得可以通过向量不经意线性评估 (vector oblivious linear evaluation, VOLE) 高效实现 OPE. 具体而言, 本文通过随机数的引入, 将高阶多项式

表 1　基于不同原语的不经意编码协议比较. n 为编码数据集的大小, ϵ 表示 **OKVS** 的扩展因子, 通常介于 **0.2** 和 **0.4** 之间.

Table 1　Comparison of oblivious encoding protocols based on different basic primitives. n is the size of the encoded dataset, and ϵ represents the expansion factor of OKVS, typically ranging between 0.2 and 0.4.

Cryptographic primitive	Computational complexity	Communication complexity	Storage complexity	Round
PNI	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	6
OKVS	$\mathcal{O}(n\log n)$	$\mathcal{O}((1+\epsilon)n)$	$\mathcal{O}((1+\epsilon)n)$	6
OPE	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	2

分解为自变量的幂次项与多个一次多项式的乘积. 自变量的幂项可以由 P_0 本地计算, 从而避免了基于密文数据的高次计算, 减少了不经意评估的计算开销; 多个一次多项式的不经意评估可由 VOLE 高效实现. 因此, 本文将 OPE 转化为多个线性多项式的不经意评估问题. 本文在表 1 中总结并比较了基于不同原语的不经意编码方法的计算、通信、存储开销复杂度及通信轮数.

交集成员检测与共享. 现有的交集成员检测与共享主要通过多次调用相等性测试方案实现, 其计算和通信开销随着单次相等性测试的开销线性增长. 在目前最优的相等性测试方法 [25] 中, P_1 构造了一个查找表 $\vec{T}$, 其中仅将第 y 位置设置为 1, 其他位置均为 0. 然后, P_0 将其比较数据 x 作为索引对该表进行不经意检索, 并将检索结果进行布尔共享, 实现 x 与 y 的相等检测.

本文将批量相等检测转化为使用单一索引的不经意检索问题, 从而通过仅一轮是通信开销实现交集成员检测与共享, 并实现在线阶段 $\mathcal{O}(n)$ 的通信复杂度. 具体而言, 本文通过计算 P_0 比较元素之间的差, 并对 P_1 的查找表进行不经意旋转, 确保在批量的相等检测中, 多个查找表对应的不同比较数据的相等检测结果出现在相同位置, 从而避免了重复调用不经意检索方案对查找表进行逐一检索.

本文的关键贡献总结如下.

• **通用隐私集合求交框架.** 本文分析并总结了现有的电路隐私集合求交方案, 提出了一个 CPSI 的计算框架, 其包含 3 个关键组件: 数据集本地分类、分类数据的不经意编码及交集成员检测与共享.

• **基于不经意多项式评估的不经意编码.** 本文提出了一种基于不经意多项式评估的不经意编码方案, 通过将不经意多项式评估转化为线性多项式的隐私计算, 在不增加存储开销的前提下达到线性计算复杂度.

• **批量相等检测.** 本文基于批量相等检测提出了一种高效的交集成员检测与共享方案, 通过将批量相等检测问题转化为使用一个索引的不经意检索问题, 实现了常数轮通信复杂度.

• **安全性与性能评估.** 本文对提出的协议进行了形式化的安全性证明, 并且进行了严格的性能评估. 结果表明, 本文的 CPSI 方案比目前最优的方案快 1.37 倍, 同时其通信开销仅为后者的 52%.

2　相关工作

2.1　隐私集合求交

电路隐私集合求交使得参与方基于交集元素进行任意对称函数的计算, 包括计算交集个数 [12,26] 和交集数据 [13,27] 以及基于阈值的交集计算 [14,15]. 首个 CPSI 协议采用了类似混淆电路 [28~30] 和 GMW 协议 [23] 等安全多方计算通用实现方式, 随后的一些研究对其计算和通信开销进行了优化 (参见文献 [16,17,23,24]).

Pinkas 类 [23,24]. P_0 和 P_1 首先分别对其隐私数据集进行本地映射和存储. 具体而言, P_0 基于布谷鸟哈希 (Cuckoo Hashing) 将其集合 X 映射并存储到哈希表 T_0 中, 哈希表 T_0 的每个桶仅包含一个元素. 同时, P_1 基于朴素哈希 (simple Hashing) 将其集合 Y 映射并存储到哈希表 T_1 中, 哈希表 T_1 每个桶通常包含约 $\log n_1$ 个元素. T_0 和 T_1 都由 β 个桶组成. 如果 $x_i \in X \cap Y$ 被映射到桶 $T_0[i]$ 中, 那么对应的 y 一定存储在 $T_1[i]$ 中. 接着, P_0 和 P_1 对每个桶调用批量不经意可编程伪随机函数 (batch oblivious programmable pseudo-random function, OPPRF). 如果 $x_i \in X \cap Y$, 则双方对第 i 个桶获得相同的随机数; 否则, 他们分别收到两个无关的随机数. 具体而言, P_0 和 P_1 对每个桶调用不经意伪随机函数 (oblivious pseudo-random function, OPRF) 协议, 其

中 P_0 对每个元素获得伪随机函数值 $F(k_i, x_i)$, 而 P_1 仅接收到 k_i, 但其可以本地计算所有元素的伪随机函数值. 此外, P_1 基于 $(y_i^j, F(k_i, y_i^j) \oplus t_i)$ 构造插值多项式 p 并将其发送给 P_0, 其中 t_i 是随机值. 因此, P_0 本地计算 $s_i = F(k_i, x_i) \oplus p(x_i)$, 如果 x_i 为交集元素, 则 $s_i = t_i$. 最后, P_0 和 P_1 基于 s_i 和 t_i 对每个桶调用相等检测方案. 如果 $x_i \in X \cap Y$, 则 P_0 和 P_1 获得 1 的布尔共享值, 否则获得 0 的布尔共享值. 此类方案存在计算瓶颈的主要原因是, 在调用 OPPRF 过程中, P_1 需要基于每个元素及其对应的为随机函数值进行多项式插值生成高次多项式, 这使得方案的计算复杂度为 $\mathcal{O}(n^2)$.

Rindal 类 [17]. 此方案的设计思路与 Pinkas 类 [24] 基本一致. 不同之处在于其基于 OKVS 构造了 OPPRF, 从而优化了 Pinkas 类中插值多项式的引入的平方级计算开销. 具体地, P_1 将键值对 $(y_i^j, z_i - F(k_i, y_i^j))$ 编码为向量 $\vec{P}$ 并发送给 P_0, 其中 z_i 是均匀随机的. 随后, P_0 计算 $x_i^* := F(k_i, x_i) + \mathsf{Decode}(\vec{P}, x_i)$. 对于交集元素 x_i, 解码计算满足 $\mathsf{Decode}(\vec{P}, x_i) = z_i - F(k_i, y_i^j)$, 即 $x_i^* = z_i$. 因此, P_0 和 P_1 对交集元素获得相同的随机值. 尽管 OKVS 的引入将 OPPRF 的计算复杂度优化至 $\mathcal{O}(n \log n)$, 但其编码过程引入了辅助扩展向量, 导致方案增加了约 20%∼40% 的存储开销.

Chandran 类 [16]. 与之前的工作 [17,23,24] 类似, P_0 和 P_1 首先分别基于布谷鸟哈希和朴素哈希对其隐私数据集进行本地映射和存储, 并生成哈希表 T_0 和 T_1, 其中 P_1 的每一个元素 y_i 被映射到 T_1 的 3 个桶中. 因此, 文献 [16] 相较先前工作的不同之处在于, 其构造了一个宽松批量 OPPRF (relaxed batch OPPRF, RB-OPPRF), 确保存储交集元素 y_i 的桶对应的随机数一定包含存储 x 的桶对应的随机数. 然后, 参与方可通过隐私集合成员检测 (private set membership, PSM) 实现交集结果的共享. 在 RB-OPPRF 的实现过程中, P_1 构造了一个额外的混淆哈希表对每个 y_i 的映射位置进行存储. 此外, P_1 将 $F(k_i, y_i)$ 拆分为三部分, 结合随机数为每个桶生成掩码, 从而减少了 OPPRF 编码值的长度. 尽管 RB-OPPRF 的构造优化了批量 OPPRF 的计算开销, 但混淆哈希表的构造进一步增加了方案的存储开销. 此外, 与先前相等检测比较两个数据是否相等不同的是, PSM 用于判断一个数据值是否属于一个集合 (判断存储每一个数据 x_i 的桶对应的随机数是否属于 T_1 的 3 个桶对应的随机数). 相较于相等检测, PSM 引入了额外的计算和通信开销.

2.2　不经意多项式评估

不经意多项式评估 [31] 是安全两方计算中的重要协议之一, 其被广泛应用于隐私集合求交 [32,33]、不经意数据过滤 (oblivious data filtering) [34]、安全关键词搜索 (secure keyword search) [35,36] 和加密查询与数据库检索等领域. OPE 在隐私集合求交中的应用主要集中于非对称数据集场景, 即 P_0 的数据集大小远远小于 P_1 的数据集大小. 由于现有的不经意多项式评估协议主要基于同态加密实现, 因此高次多项式昂贵的计算开销使其无法适用于数据量较大的对称数据集. 基于同态加密的不经意多项式评估构造 [32,33] 如下. P_0 首先对 x 进行加密, 即 $[x] := HE.Enc(x)$, 并将其发送给 P_1. 随后, P_1 基于同态密文进行多项式计算, 得到 $[f(x)] := a_n[x^n] + \cdots + a_1[x] + a_0$, 并将结果返回给 P_0. 最后, P_0 解密 $[f(x)]$ 得到 $f(x)$. 现有的优化思路主要在于提高基于同态密文的高次多项式计算效率. 然而, 对于对称数据集的隐私集合求交方案, 基于同态的不经意多项式评估仍然存在显著的性能瓶颈.

3　预备知识

3.1　符号定义

本文用 $[m]$ 表示索引集合 $\{1, \ldots, m\}$, 用 $[a, b]$ 表示索引集合 $\{a, a+1, \ldots, b\}$. 将向量 $a_0, \ldots, a_{n-1}$ 表示为 $\vec{A}$, 并定义 a_i 为 $\vec{A}$ 的第 i 个元素. 将向量 $\vec{A}$ 与 $\vec{B}$ 的内积表示为 $\vec{A} \cdot \vec{B}$. 本文用 $\mathsf{rotate}(\vec{A}, r)$ 表示将向量 $\vec{A}$ 逆时针旋转 r 次. $\mathbf{1}\{b\}$ 为指示函数, 当 b 为真时其值为 1; 否则, 其值为 0. 本文用 $\mathbf{M}_j$ 表示矩阵 $\mathbf{M}$ 的第 j 列.

3.2　安全模型

本文所提协议实现了标准的半诚实模型下的安全性. 在该模型下, 攻击者可能尝试从合法的消息中获取隐私信息, 但其必须严格遵循协议的执行流程. 本文工作的安全性证明基于通用可组合性 (universal composability,

UC) 框架 [37], 该框架采用基于模拟的安全范式. 在 UC 框架中, 协议在多个互联的机器间执行. 网络攻击者 $\mathcal{A}$ 可以控制部分未被攻破机器的通信带 (communication tapes), 即观察未被攻破参与方的收/发消息, 并影响消息的发送顺序. 如果对于任意一个试图攻击协议 Π 的概率多项式时间 (probabilistic polynomial time, PPT) 敌手 $\mathcal{A}$, 都存在一个试图攻击理想功能函数 $\mathcal{F}$ 的 PPT 模拟器 $\mathcal{S}$, 使得 $\mathcal{Z}$ 无法区分执行协议 Π 的 $\mathcal{A}$ 与执行理想函数 $\mathcal{F}$ 的 $\mathcal{S}$, 则称协议 Π UC 安全地实现了功能函数 $\mathcal{F}$.

理想世界执行 $\mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda)$**.** 在理想世界中, 参与方 $\mathcal{P} := P_0, P_1$ 仅与理想功能函数 ($\mathcal{F}_{\mathsf{OPE}}$, $\mathcal{F}_{\mathsf{batch\text{-}eq}}$) 进行交互. 各方将其输入发送给理想功能函数, 完成计算后理想功能函数将结果返回给 P_0 和 P_1.

真实世界执行 $\mathsf{Real}_{\Pi,\mathcal{A},\mathcal{Z}}(1^\lambda)$**.** 在真实世界中, 参与方 $\mathcal{P} := P_0, P_1$ 彼此之间直接通信, 并共同执行协议 Π.

定义1　如果对于所有 PPT 敌手 $\mathcal{A}$, 都存在一个 PPT 模拟器 $\mathcal{S}$, 使得对于所有 PPT 环境 $\mathcal{Z}$, 均满足式 (1), 则称协议 Π UC 安全地实现了功能函数 $\mathcal{F}$,

$$\mathsf{Real}_{\Pi,\mathcal{A},\mathcal{Z}}(1^\lambda) \approx \mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda). \tag{1}$$

3.3　基础组件

向量不经意线性评估. 向量不经意线性评估 [38] 是不经意传输 (oblivious transfer) 在算术域中的扩展. 在向量不经意线性评估中, 发送方 P_1 输入一对向量, 而接收方 P_0 获得这对向量的某一线性组合. 正式地, 给定一个有限域 $\mathbb{F}$, 接收方 P_0 向 VOLE 的理想函数 $\mathcal{F}_{\mathsf{VOLE}}$ 输入一个标量 $x \in \mathbb{F}$, 发送方 P_1 输入一对向量 $(\vec{U}, \vec{V}) \in \mathbb{F}^n \times \mathbb{F}^n$. 随后 $\mathcal{F}_{\mathsf{VOLE}}$ 计算并输出 $\vec{W} := \vec{U}x + \vec{V}$ 给 P_0. 在此过程中, P_0 无法获得关于 $(\vec{U}, \vec{V})$ 的任何信息, 且 P_1 无法获知 x 的值, 其功能函数如算法 1 所示.

算法 1 向量不经意线性评估的功能函数 $\mathcal{F}^n_{\mathsf{VOLE}}[x, \vec{U}, \vec{V}]$.

输入: 只要收到 P_0 发送的 $(\mathsf{Input}, \mathsf{sid}, x)$, 记录 x, 并将 $(\mathsf{Input}, \mathsf{sid}, P_0)$ 发送给 $\mathcal{S}$; 只要收到 P_1 发送的 $(\mathsf{Input}, \mathsf{sid}, (\vec{U}, \vec{V}))$, 记录 $(\vec{U}, \vec{V})$, 并将 $(\mathsf{Input}, \mathsf{sid}, P_1)$ 发送给 $\mathcal{S}$;
执行: 如果 x, $\vec{U}$ 和 $\vec{V}$ 均被记录, 计算 $\vec{W} := \vec{U}x + \vec{V}$;
输出: 将 $(\mathsf{Output}, \mathsf{sid}, \vec{W})$ 发送给 P_0, P_1 无输出.

随机向量不经意偏移评估. 随机向量不经意偏移评估 (vector oblivious shift evaluation, RVOSE) [25] 是基于向量计算的随机 VOLE 的扩展. 对于 RVOSE 的功能函数 $\mathcal{F}_{\mathsf{RVOSE}}$, 参与方 P_0 和 P_1 都没有输入. $\mathcal{F}_{\mathsf{RVOSE}}$ 执行后, P_0 收到一个标量 ε 和一个向量 $\vec{W}$, P_1 收到一组向量 $(\vec{U}, \vec{V})$, 满足 $\vec{W} = \mathsf{rotate}(\vec{U}, \varepsilon) \oplus \vec{V}$. 在此过程中, P_0 无法获取 $(\vec{U}, \vec{V})$ 的任何信息, P_1 也无法获取 $(\varepsilon, \vec{W})$ 的任何信息, 其功能函数如算法 2 所示.

算法 2 随机向量不经意偏移评估的功能函数 $\mathcal{F}^n_{\mathsf{RVOSE}}$.

输入: 一旦协议开始, 将 $(\mathsf{Input}, \mathsf{sid}, P_0)$ 和 $(\mathsf{Input}, \mathsf{sid}, P_1)$ 发送给 $\mathcal{S}$;
执行: 随机采样 $\varepsilon \in [0, n-1]$ $(\vec{U}, \vec{V} \in \mathbb{Z}^n)$, 并计算 $\vec{W} = \mathsf{rotate}(\vec{U}, \varepsilon) \oplus \vec{V}$;
输出: 将 $(\mathsf{Output}, \mathsf{sid}, (\varepsilon, \vec{W}))$ 发送给 P_0, 将 $(\mathsf{Output}, \mathsf{sid}, (\vec{U}, \vec{V}))$ 发送给 P_1.

相等检测. 对于相等检测 [39] 的功能函数 $\mathcal{F}_{\mathsf{eq}}$, P_0 输入一个整数 $x \in \{0,1\}^n$, P_1 输入一个整数 $y \in \{0,1\}^n$. 随后 $\mathcal{F}_{\mathsf{eq}}$ 将 $\mathbf{1}\{x = y\}$ 的布尔共享发送给 P_0 和 P_1, 当且仅当 $x = y$ 时该值为 1, 否则为 0. 在此过程中, P_0 无法获知关于 y 的任何信息, P_1 也无法获取 x 的任何信息, 其功能函数如算法 3 所示.

算法 3 相等检测的功能函数 $\mathcal{F}^n_{\mathsf{eq}}(x, y)$.

输入: 只要收到 P_0 发送的 $(\mathsf{Input}, \mathsf{sid}, x)$, 记录 x, 并将 $(\mathsf{Input}, \mathsf{sid}, P_0)$ 发送给 $\mathcal{S}$; 只要收到 P_1 发送的 $(\mathsf{Input}, \mathsf{sid}, y)$, 记录 y, 并将 $(\mathsf{Input}, \mathsf{sid}, P_1)$ 发送给 $\mathcal{S}$;
执行: 如果 x, y 均被记录, 计算 $[e]_0 \oplus [e]_1 := \mathbf{1}\{x = y\}$, 当且仅当 $x = y$, $[e]_0 \oplus [e]_1 = 1$, 否则 $[e]_0 \oplus [e]_1 = 0$;
输出: 将 $(\mathsf{Output}, \mathsf{sid}, [e]_0)$ 发送给 P_0, 将 $(\mathsf{Output}, \mathsf{sid}, [e]_1)$ 发送给 P_1.

4　基于向量不经意线性评估的不经意多项式评估

摒弃基于同态加密隐私计算高阶多项式的思路, 本文将 OPE 转化为高阶多项式分解问题, 从而基于 VOLE 高效实现 OPE, 其功能函数如算法 4 所示.

算法 4 不经意多项式评估的功能函数 $\mathcal{F}_{\mathsf{OPE}}^{n}[x, f(y)]$.

输入: 只要收到 P_0 发送的 $(\mathsf{Input}, \mathsf{sid}, x)$, 记录 x, 并将 $(\mathsf{Input}, \mathsf{sid}, P_0)$ 发送给 $\mathcal{S}$; 只要收到 P_1 发送的 $(\mathsf{Input}, \mathsf{sid}, f(\cdot))$, 记录 $f(\cdot)$, 并将 $(\mathsf{Input}, \mathsf{sid}, P_1)$ 发送给 $\mathcal{S}$, 其中 $f(y) := a_n y^n + \cdots + a_1 y + a_0$;

执行: 如果 $x, f(\cdot)$ 均被记录, 计算 $f(x) := a_n x^n + \cdots + a_1 x + a_0$;

输出: 将 $(\mathsf{Output}, \mathsf{sid}, f(x))$ 发送给 P_0.

4.1　协议概述

多项式分解. 对于一个以 x 为自变量的高阶多项式, 本文可以递归地将其分解为一个高阶向量与一个线性多项式向量的内积. 具体而言, 一个 n 阶多项式 $f(x)$ 可以通过式 (2) 进行分解, 其中 q_c 为常数项, q_x 为一个 $n-1$ 阶的多项式,

$$\begin{aligned} f(x) &:= a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \\ &= x \cdot q_x + q_c. \end{aligned} \tag{2}$$

通过递归应用此分解思路, q_x 可以进一步分解为一次线性多项式. 当 $n = 3$ 时, $f(x)$ 的分解过程如式 (3) 所示:

$$\begin{aligned} f(x) &:= a_3 x^3 + a_2 x^2 + a_1 x + a_0 \\ &= x(a_3 x^2 + a_2 x + a_1) + a_0 \\ &= x(x(a_3 x + a_2) + a_1) + a_0 \\ &= (x^2, x, 1) \cdot (a_3 x + a_2, a_1, a_0) \\ &= \vec{X} \cdot \vec{S}. \end{aligned} \tag{3}$$

基于 VOLE 的 OPE. 在计算 $f(x) := \vec{X} \cdot \vec{S}$ 的过程中, $\vec{X}$ 中的每一项均可以由 P_0 本地计算得到, 因此, 本文的目标是如何基于 VOLE 高效计算 $\vec{S}$. 然而, $\vec{S}$ 的构造并不完全契合 VOLE 的输入结构, 因此本文无法直接将 $\vec{S}$ 分解为 $\vec{U}$ 和 $\vec{V}$, 使得 $\vec{S} = \vec{U}x + \vec{V}$. 为了解决这一问题, 本文在多项式分解中引入了随机数, 将 x 的每一项重构为一次线性多项式, 使得 $\vec{S}$ 可以通过 VOLE 高效计算. 式 (4) 展示了当 $n = 3$ 时引入随机数后 $\vec{S}$ 的分解过程,

$$\begin{aligned} f(x) &:= a_3 x^3 + a_2 x^2 + a_1 x + a_0 \\ &= x(a_3 x^2 + a_2 x + a_1) + a_0 \\ &= x(a_3 x^2 + a_2 x + (a_1 - r_1 + r_1)) + a_0 \\ &= x(a_3 x^2 + a_2 x + (a_1 - r_1)) + (r_1 x + a_0) \\ &= x(x(a_3 x + (a_2 - r_2 + r_2)) + (a_1 - r_1)) + (r_1 x + a_0) \\ &= x(x(a_3 x + (a_2 - r_2)) + (r_2 x + (a_1 - r_1))) + (r_1 x + a_0) \\ &= (x^2, x, 1) \cdot (a_3 x + (a_2 - r_2), r_2 x + (a_1 - r_1), r_1 x + a_0) \\ &= \vec{X} \cdot \vec{W}. \end{aligned} \tag{4}$$

显然, 如式 (5) 所示, 当 $f(x)$ 的最高次项扩展为 x^n 时, $\vec{W}$ 可以快速分解为 $\vec{U}$ 和 $\vec{V}$, 使得 $\vec{W} = \vec{U}x + \vec{V}$. 因此,

$\vec{W}$ 的计算可通过 $\mathcal{F}_{\mathsf{VOLE}}$ 高效实现,

$$\vec{W} = \begin{bmatrix} a_n \\ \vdots \\ r_2 \\ r_1 \end{bmatrix} x + \begin{bmatrix} a_{n-1} - r_{n-1} \\ \vdots \\ a_1 - r_1 \\ a_0 \end{bmatrix} = \vec{U}x + \vec{V}. \tag{5}$$

4.2　协议详细描述

本文的协议 $\Pi^n_{\mathsf{OPE}}(X, Y)$ 的流程描述如下.

(1) 线性多项式向量的本地构造. 第一步, P_1 生成随机数 r_i, 其中 $i \in [1, n-1]$. 然后 P_1 基于 r_i 计算 $\vec{U} := (a_n, r_{n-1}, \ldots, r_1)$ 和 $\vec{V} := (a_{n-1} - r_{n-1}, a_{n-2} - r_{n-2}, \ldots, a_0)$.

(2) VOLE 调用. 第二步, P_0 和 P_1 对 $\mathcal{F}_{\mathsf{VOLE}}$ 进行调用. 具体地, P_0 输入 x, P_1 输入 $(\vec{U}, \vec{V})$. $\mathcal{F}_{\mathsf{VOLE}}$ 执行后, P_0 接收到 $\vec{W}$, 其中 $\vec{W} := \vec{U} \cdot x + \vec{V}$.

(3) 多项式评估. 第三步, P_0 通过本地计算完成不经意多项式评估. 具体而言, P_0 首先计算 x^i, 其中 $i \in [1, n-1]$, 并构造 $\vec{X} := (x^{n-1}, \ldots, x, 1)$. 然后, P_0 计算 $f(x) = \vec{X} \cdot \vec{W}$.

效率分析. 在协议 $\Pi^n_{\mathsf{OPE}}(X, Y)$ 中, 第一和三步分别由 P_1 和 P_0 在本地计算. 因此, 基于 VOLE 的 OPE 的通信开销仅由第二步决定, 即调用一次 VOLE 的通信开销, 其与 $\vec{U}$ 和 $\vec{V}$ 向量的维数呈线性关系.

4.3　安全性证明

本文证明协议 Π_{OPE} 安全实现了功能函数 $\mathcal{F}_{\mathsf{OPE}}$.

定理1　协议 Π_{OPE} 在 $\mathcal{F}_{\mathsf{VOLE}}$- 混合模型下, 针对静态入侵假设下的半诚实 PPT 对手, UC 安全地实现了功能函数 $\mathcal{F}_{\mathsf{OPE}}$.

证明　为了证明定理 1, 本文构造了一个 PPT 模拟器 $\mathcal{S}$, 使得不存在 PPT 环境 $\mathcal{Z}$ 能够区分理想世界 $\mathsf{Ideal}_{\mathcal{F}_{\mathsf{OPE}},\mathcal{S},\mathcal{Z}}(1^\lambda)$ 和真实世界 $\mathsf{Real}^{\mathcal{F}_{\mathsf{VOLE}}}_{\Pi_{\mathsf{OPE}},\mathcal{A},\mathcal{Z}}(1^\lambda)$. 本文考虑以下几种实例.

实例 1　P_0 被敌手控制. 本文构造模拟器 $\mathcal{S}$ 模拟 $\mathcal{F}_{\mathsf{VOLE}}$ 和诚实参与方 P_1, 并对 $\mathcal{Z}$ 发送/接收的消息进行转发, 其中 $\mathcal{S}$ 内部运行着 $\mathcal{A}$.

在 $\mathcal{F}_{\mathsf{OPE}}$ 接收到 $(\mathsf{Input}, \mathsf{sid}, P_1)$ 后, $\mathcal{S}$ 执行以下操作.

• 当被敌手控制的参与方 P_0 向 $\mathcal{F}_{\mathsf{VOLE}}$ 输入 x 时, $\mathcal{S}$ 记录 x 并计算 $\vec{X} := (x^{n-1}, \ldots, x^2, x, 1)$.

• 在接收到 $\mathcal{F}_{\mathsf{OPE}}$ 的输出 $(\mathsf{Output}, \mathsf{sid}, f(x))$ 后, $\mathcal{S}$ 随机采样 r_i 并计算 $r_1 := f(x) - r_2 x + \cdots + r_n x^{n-1}$, 其中 $i \in [2, n]$.

• $\mathcal{S}$ 模拟 $\mathcal{F}_{\mathsf{VOLE}}$ 将 $\vec{W} := (r_n, \ldots, r_1, r_0)$ 发送给 P_0.

不可区分性. 本文证明理想世界中输入的消息和 P_0 的输出与真实世界中的消息和输出是不可区分的.

断言1　理想世界 $\mathsf{Ideal}_{\mathcal{F}_{\mathsf{OPE}},\mathcal{S},\mathcal{Z}}(1^\lambda)$ 和真实世界 $\mathsf{Real}^{\mathcal{F}_{\mathsf{VOLE}}}_{\Pi_{\mathsf{OPE}},\mathcal{A},\mathcal{Z}}(1^\lambda)$ 是完美不可区分的.

证明　在理想世界中, $\vec{W}$ 中的每个元素都是均匀随机的. 在真实世界中, $\vec{W}$ 中的每个元素基于 $\vec{U}x + \vec{V}$ 计算生成. 然而, $\vec{V}$ 的第 $i+1$ 个元素是 $a_i - r_i$, 其中 $i \in [2, n]$, 而 $\vec{U}$ 的第一个元素是 r_1. 因此, $\vec{U}x + \vec{V}$ 中的每个元素都是一个随机值. 因此, $\vec{W}$ 中的每个元素在理想世界和真实世界中都是均匀随机的.

实例 2　P_1 被敌手控制. 本文构造模拟器 $\mathcal{S}$ 模拟 $\mathcal{F}_{\mathsf{VOLE}}$ 和诚实参与方 P_0, 并对 $\mathcal{Z}$ 发送/接收的消息进行转发, 其中 $\mathcal{S}$ 内部运行着 $\mathcal{A}$.

在协议 Π_{OPE} 中, P_1 不会接收到来自 P_0 的任何消息. 因此, P_1 无法区分理想世界和真实世界.

5　批量相等检测

摒弃反复调用相等检测协议来实现批量相等检测的思想, 本文通过索引归一化的思想将批量相等检测问题转换为使用单一索引的不经意检索问题. 批量相等检测的功能函数如算法 5 所示.

算法 5 批量相等检测的功能函数 $\mathcal{F}_{\mathsf{batch\text{-}eq}}^{m,\log n}[X,Y]$.

输入: 只要收到 P_0 发送的 $(\mathsf{Input},\mathsf{sid},X)$, 记录 X, 并将 $(\mathsf{Input},\mathsf{sid},P_0)$ 发送给 $\mathcal{S}$, 其中 $|X|=m$ 且 $x_i\in\{0,1\}^{\log n}$; 只要收到 P_1 发送的 $(\mathsf{Input},\mathsf{sid},Y)$, 记录 Y, 并将 $(\mathsf{Input},\mathsf{sid},P_1)$ 发送给 $\mathcal{S}$, 其中 $|Y|=m$ 且 $y_i\in\{0,1\}^{\log n}$;

执行: 如果 X,Y 均被记录, 计算 $\vec{E}:=\{E_i\mid i\in[1,m]\}$, 其中 $E_i=[E_i]_0\oplus[E_i]_1=\mathbf{1}\{x_i=y_i\}$;

输出: 将 $(\mathsf{Output},\mathsf{sid},[\vec{E}]_0)$ 发送给 P_0, 将 $(\mathsf{Output},\mathsf{sid},[\vec{E}]_1)$ 发送给 P_1, 其中 $[\vec{E}]_j:=\{[E_i]_j\mid i\in[1,m]\}$, $j\in\{0,1\}$.

5.1　协议概述

P_0 和 P_1 需要对 m 组数据 (x_i,y_i) 执行批量相等检测协议, 其中 $i\in[m]$, $x_i\in\{0,1\}^{\log n}$. 一种不考虑安全性的简单思想如下所示. P_1 对每个 y_i 本地生成一个长度为 n 的查找表, 其中仅第 y_i 个位置为 1, 其他位置均为 0. 然后, P_1 以每个查找表为列, 生成一个矩阵 $\mathbf{M}$, 其行数为 n, 列数为 m. 在此基础上, 本文的思路是将矩阵 $\mathbf{M}$ 的每一列进行旋转, 使得 m 组相等检测的结果落在矩阵 $\mathbf{M}$ 的同一行, 使得可以通过一不经意检索获得 m 组相等检测的结果.

不安全的批量相等检测方案. 一种简单但不考虑安全性的实现方式是通过泄露 x_i 之间的差, 使得 P_1 对 $\mathbf{M}$ 的每一列进行本地旋转. 具体地, P_0 计算 $o_i:=x_1-x_i$ $(i\in[2,m])$, 并将其发送给 P_1; 然后 P_1 将 o_i 作为偏移量对 $\mathbf{M}$ 的第 i 列进行旋转, 使得所有 x_i 与 y_i 的相等检测结果落到 $\mathbf{M}$ 的第 x_1 行. 然而, 这种简单的方法面临两个挑战: 首先, 它直接给 P_1 泄露了 x_1 和所有 x_1-x_i $(i\in[2,m])$; 其次, 它将所有批量相等检测的结果泄露给 P_0, 而不是在 P_0 和 P_1 之间安全地共享.

安全但不高效的批量相等检测方案. 对于将第二点挑战的安全需求总结如下. P_0 拥有偏移量 o_i, 其中 $i\in[2,m]$, P_1 拥有一个二进制向量 $\vec{T_i}$ (即 $\mathbf{M}$ 的第 i 列). 它们基于偏移量 o_i 对 $\vec{T_i}$ 进行旋转, 使得旋转后的向量在 P_0 和 P_1 之间共享. 在此过程中, P_1 无法获得 o_i 的相关信息, P_0 无法获得 $\vec{T_i}$ 的相关信息. 上述隐私需求可以基于标准的 VOSE 协议实现.

随机 *VOSE* 向标准 *VOSE* 的转换. 在离线阶段, P_0 和 P_1 调用 RVOSE, 然后 P_1 获得两个随机二进制向量 $\vec{U_i}$ 和 $\vec{V_i}$, P_0 获得一个随机偏移量 ε_i 和一个向量 $\vec{W_i}$, 满足 $\vec{W}:=\mathsf{rotate}(\vec{U_i},\varepsilon_i)\oplus\vec{V_i}$. 在在线阶段, P_0 和 P_1 将偏移量和旋转向量从随机值转换为指定值. 具体流程如下.

• 对于偏移量从随机值转换为指定值: P_0 将 $o_i-\varepsilon_i$ 发送给 P_1. 然后, P_0 将 $\vec{W_i}$ 基于偏移量 $o_i-\varepsilon_i$ 进行旋转, 生成 $\vec{W_i'}$; P_1 基于偏移量 $o_i-\varepsilon_i$ 对 $\vec{V_i}$ 进行本地旋转, 生成 $\vec{V_i'}$. 因此, $\vec{W_i'}=\mathsf{rotate}(\vec{U_i},o_i)\oplus\vec{V_i'}$, 即偏移量从 ε_i 转换为 o_i.

• 对于旋转向量从随机值转换为指定值: P_1 计算 $\vec{S_i}:=\vec{T_i}\oplus\vec{U_i}$ 并将其发送给 P_0, 然后 P_0 本地计算 $\vec{S_i'}:=\mathsf{rotate}(\vec{S_i},o_i)\oplus\vec{W_i'}$.

• $\vec{S_i'}$ 和 $\vec{V_i'}$ 为将 $\vec{T_i}$ 基于偏移量 o_i 旋转之后的共享结果, 由 P_0 和 P_1 分别持有.

在上述过程中, P_1 在在线阶段将 m 个向量发送给 P_0, 而 P_0 将 m 个索引作为偏移量发送给 P_1, 因此, 此阶段的通信开销为 $(n+\log n)\cdot m$ 比特.

性能优化. 为了进一步减少在线阶段的通信开销, 本文将 P_1 发送给 P_0 的向量压缩为一个索引值. 具体地, P_0 和 P_1 在离线阶段旋转一个与在线阶段传输的 $\vec{T_i}$ 具有相同特性的向量, 其满意只有一个位置 (g_i) 的值为 1, 所有其他位置的值为 0. 与在线阶段不同的是, g_i 是随机确定的, 而不是由 P_1 指定. 因此, 在在线阶段, 本文的目标是将 x_i 和 y_i 之间的相等检测转换为 g_i 与另一个随机值 h_i 的比较. h_i 的推导过程如下. 当且仅当 $x_i=y_i$, 有 $x_i+r_i=y_i+r_i$. 当 $r_i=g_i-y_i$ 时, $y_i+r_i=g_i$. 因此, $x_i+r_i=x_i+g_i-y_i$, 从而得到 $h_i=x_i+g_i-y_i$. 为了让 P_0 获得 h_i $(i\in[1,m])$, P_1 计算 $r_i=g_i-y_i$ 并将其发送给 P_0. 为了将批量相等检测的结果旋转到同一行, P_0

计算 $o_i := h_1 - h_i + \varepsilon_1 - \varepsilon_i$ 并将其发送给 P_1, 其中 $i \in [2, m]$. 然后, P_0 计算检索索引 $\mathsf{ind} := h_1 + \varepsilon_1$ 发送给 P_1. 因此, 双方通过在本地选择共享矩阵的第 ind 行, 可得到相等检测的结果. 通过上述操作将批量相等检测的在线阶段通信开销减少到 $2m \cdot \log n$ 比特.

直观地, 目前的协议似乎需要两轮通信. 然而, 通过分析已知 $o_i = (x_1 - x_i + \varepsilon_1 - \varepsilon_i) + (r_1 - r_i) = (k_1 - k_i) + (r_1 - r_i)$, 且检索索引 $h_1 + \varepsilon_1 = k_1 + r_1$, 其中 $k_i = x_i + \varepsilon_i$, $r_i = g_i - y_i$. 因此, 通过让 P_0 和 P_1 同时交换 k_i 和 r_i ($i \in [1, m]$), 他们可以在本地计算 o_i, h_i 和 ind, 将在线阶段的通信轮数减少为一轮.

基于差分泄露的性能优化. 在上述协议中, 对于 $i \in [1, m]$, P_0 和 P_1 执行 m 次 Π_{RVOSE} 对 o_i 和 g_i 进行隐藏. 然而, 通过分析 o_i 和 k_i 的生成过程可知, 如果在离线阶段仅执行一次 Π_{RVOSE}, 且在在线阶段 P_0 计算 $k_i = x_i + \varepsilon_1$, 则 P_1 将获得不同 x_i 之间的差. 由于 ε_1 的随机性, P_1 无法推导出具体的 x_i 值. 因此, 在允许差分泄露的安全模型下, 通过 $x_i - x_j$ 的泄露 ($i \neq j$), 可以通过一次 Π_{RVOSE} 执行来实现批量相等, 从而降低协议的整体通信开销. 此优化思路仅适用于对差分泄漏不敏感的隐私计算场景. 例如, 在电子投票隐私校验系统中, 系统仅在意选票结果是否正确, 而不在意选票之间的差分; 在区块链智能合约中的条件验证中, 合约执行方仅在意一致性验证的结果等.

5.2　协议详细描述

本文所提方案 $\Pi^{m,\log n}_{\mathsf{batch\text{-}eq}}(X, Y)$ 描述如下.

在离线阶段:

(1) RVOSE 的调用. 第一步, P_0 和 P_1 调用 Π_{RVOSE} 并分别获得 $(\vec{W}_i, \varepsilon_i)$ 和 $(\vec{U}_i, \vec{V}_i)$, 其中 $\vec{W} = \mathsf{rotate}(\vec{U}_i, \varepsilon_i) \oplus \vec{V}_i$, $i \in [m]$.

(2) 将 RVOSE 转为标准 VOSE (向量转换). 对于 $i \in [m]$, P_1 首先生成一个二进制向量 $\vec{T}'_i \in \mathbb{Z}_2^n$, 其中仅 $T'_i[g_i] = 1$, 其他位置为 0. 然后, P_1 计算 $\vec{S}_i := \vec{T}'_i \oplus \vec{U}_i$ 并将其发送给 P_0.

(3) 查找表的预共享. 对于 $i \in [m]$, P_0 计算 $\vec{W}'_i := \mathsf{rotate}(\vec{S}_i, \varepsilon_i) \oplus \vec{W}_i$.

在在线阶段:

(1) 将 RVOSE 转为标准 VOSE (索引转换). 对于 $i \in [m]$, P_0 计算 $k_i := x_i + \varepsilon_i$, P_1 计算 $r_i := g_i - y_i$. 它们同时交换 k_i 和 r_i. 然后, P_0 计算 $h_i := x_i + r_i$.

(2) 检索索引的归一化. 对于 $i \in [2, m]$, P_0 和 P_1 在本地计算 $o_i := (k_1 - k_i) + (r_1 - r_i)$.

(3) 查找表的共享. 对于 $i \in [2, m]$, P_0 和 P_1 生成 $[\vec{M}_i]_0 := \mathsf{rotate}(\vec{W}'_i, o_i)$ 和 $[\vec{M}_i]_1 := \mathsf{rotate}(\vec{V}_i, o_i)$. 然后, 它们分别使用 $[\vec{M}_i]_j$ 作为列向量生成矩阵 $\mathbf{M}_j$, 其中 $j \in \{0, 1\}$.

(4) 检索索引的恢复. 对于 $j \in \{0, 1\}$, P_j 在本地计算 $\mathsf{ind} := k_1 + r_1$ 作为选择索引.

(5) 结果检索. P_0 和 P_1 分别选择 $\mathbf{M}_j$ 的第 ind 行, 记为 $[\vec{E}]_j$, 作为批量相等检测的结果.

效率分析. 在离线阶段, P_0 和 P_1 调用 m 次 Π_{RVOSE}, 每次调用的通信开销为 $\lambda \log n$ 比特 [25]. 此外, P_1 将 m 个向量 $\vec{S}_i \in \mathbb{Z}_2^n$ 发送给 P_0. 因此, 离线阶段的总通信开销为 $m\lambda \log n + mn$ 比特. 在在线阶段, P_0 和 P_1 同时交换 k_i 和 r_i, 其中 $i \in [m]$. 因此, 在线阶段的总通信开销为 $2m \log n$ 比特.

5.3　安全性证明

本文证明协议 $\Pi_{\mathsf{batch\text{-}eq}}$ 安全实现了功能函数 $\mathcal{F}_{\mathsf{batch\text{-}eq}}$.

定理2　协议 $\Pi_{\mathsf{batch\text{-}eq}}$ 针对静态入侵假设下的半诚实 PPT 敌手, 在 $\mathcal{F}_{\mathsf{RVOSE}}$- 混合模型下,UC 安全的实现了功能函数 $\mathcal{F}_{\mathsf{batch\text{-}eq}}$.

证明　为了证明定理 2, 本文构造了一个 PPT 模拟器 $\mathcal{S}$, 使得没有 PPT 环境 $\mathcal{Z}$ 能够区分理想世界 $\mathsf{Ideal}_{\mathcal{F}_{\mathsf{batch\text{-}eq}},\mathcal{S},\mathcal{Z}}(1^\lambda)$ 和真实世界 $\mathsf{Real}^{\mathcal{F}_{\mathsf{RVOSE}}}_{\Pi_{\mathsf{batch\text{-}eq}},\mathcal{A},\mathcal{Z}}(1^\lambda)$. 本文考虑以下几种实例.

实例 3　P_0 被敌手控制. 本文构造模拟器 $\mathcal{S}$ 模拟 $\mathcal{F}_{\mathsf{RVOSE}}$ 和诚实参与方 P_1, 并对 $\mathcal{Z}$ 发送/接收的消息进行转发, 其中 $\mathcal{S}$ 内部运行着 $\mathcal{A}$.

在 $\mathcal{F}_{\mathsf{batch\text{-}eq}}$ 收到 $(\mathsf{Input}, \mathsf{sid}, P_1)$ 后, $\mathcal{S}$ 执行以下操作.

• 当 P_0 调用 $\mathcal{F}_{\mathsf{RVOSE}}$ 时, $\mathcal{S}$ 为 $i \in [m]$ 随机采样 $\vec{W}_i \in \mathbb{Z}_2^n$ 和 $\varepsilon_i \in [0, n-1]$ 并将它们发送给 P_0.

• 对于 $i \in [m]$, $\mathcal{S}$ 生成二进制向量 $\vec{S}_i \in \mathbb{Z}_2^n$, 满足每个向量不是全 0 和全 1.

• $\mathcal{S}$ 计算 $\vec{W}_i' := \mathsf{rotate}(\vec{S}_i, \varepsilon_i) \oplus \vec{W}_i$, 其中 $i \in [m]$.

• 在接收到 $(\mathsf{Output}, \mathsf{sid}, [\vec{E}]0)$ 后, $\mathcal{S}$ 采样 r_{ind}, 使得 $W_1'[r_{\mathsf{ind}}] = [E_1]_0$, 并采样 r_i' $(i \in [2, m])$, 使得 $W_i'[r_i'] = [E_i]_0$.

• $\mathcal{S}$ 计算 $r_1 := r_{\mathsf{ind}} - x_1 - \varepsilon_1$.

• $\mathcal{S}$ 计算 $r_i := x_1 - x_i + \varepsilon_1 - \varepsilon_i + r_1 - r_{\mathsf{ind}} + r_i'$, 其中 $i \in [2, m]$.

• $\mathcal{S}$ 将 r_i $(i \in [m])$ 发送给 P_0.

不可区分性. 本文证明理想世界中输入的消息和 P_0 的输出与真实世界中的消息和输出是不可区分的.

断言2　理想世界 $\mathsf{Ideal}_{\mathcal{F}_{\mathsf{batch\text{-}eq}}, \mathcal{S}, \mathcal{Z}}(1^\lambda)$ 和真实世界 $\mathsf{Real}^{\mathcal{F}_{\mathsf{RVOSE}}}_{\Pi_{\mathsf{batch\text{-}eq}}, \mathcal{A}, \mathcal{Z}}(1^\lambda)$ 是完美不可区分的.

证明　$\mathsf{Ideal}_{\mathcal{F}_{\mathsf{batch\text{-}eq}}, \mathcal{S}, \mathcal{Z}}(1^\lambda)$ 和 $\mathsf{Real}^{\mathcal{F}_{\mathsf{RVOSE}}}_{\Pi_{\mathsf{batch\text{-}eq}}, \mathcal{A}, \mathcal{Z}}(1^\lambda)$ 之间, 有 3 个输入输出不同的地方.

• $\mathcal{F}_{\mathsf{RVOSE}}$ 对 P_0 的输出在理想世界和真实世界之间是不可区分的.

• 在理想世界中, $\vec{S}_i$ 和 ε_i 是随机采样的. 在真实世界中, $\vec{S}_i := \vec{T}_i' \oplus \vec{U}_i$, 其中 $\vec{T}_i'$ 由 P_1 随机采样. 因此, $\vec{S}_i$ 在理想世界和真实世界中都是均匀随机的.

• 在理想世界中, $r_1 := r_{\mathsf{ind}} - x_1 - \varepsilon_1$, 其中 r_{ind} 是均匀随机的. 对于 $i \in [2, m]$, $r_i := x_1 - x_i + \varepsilon_1 - \varepsilon_i + r_1 - r_{\mathsf{ind}} + r_i'$, 其中 ε_1 和 ε_i 是均匀随机的. 在真实世界中, $r_i := g_i - y_i$, 其中 g_i 是均匀随机的. 因此, r_i 在理想世界和真实世界中是不可区分的.

实例 4　P_1 被敌手控制. 本文构造模拟器 $\mathcal{S}$ 模拟 $\mathcal{F}_{\mathsf{RVOSE}}$ 和诚实参与方 P_1, 并对 $\mathcal{Z}$ 发送/接收的消息进行转发, 其中 $\mathcal{S}$ 内部运行着 $\mathcal{A}$.

在 $\mathcal{F}_{\mathsf{batch\text{-}eq}}$ 收到 $(\mathsf{Input}, \mathsf{sid}, P_0)$ 后, $\mathcal{S}$ 执行以下操作.

• 当 P_1 调用 $\mathcal{F}_{\mathsf{RVOSE}}$ 时, 对于 $i \in [m]$, $\mathcal{S}$ 选择 $\vec{U}_i, \vec{V}_i \in \mathbb{Z}_2^n$, 满足每个向量不是全 0 也不是全 1. 然后, $\mathcal{S}$ 将它们发送给 P_1.

• 在收到 $(\mathsf{Output}, \mathsf{sid}, [\vec{E}]_1)$ 后, $\mathcal{S}$ 随机采样 r_{ind}, 使得 $V_1[r_{\mathsf{ind}}] = [E_1]_1$, 并采样 k_i' $(i \in [2, m])$, 使得 $V_i[k_i'] = [E_i]_1$.

• 在收到 P_1 发送的 $\vec{S}_i$ $(i \in [m])$ 后, $\mathcal{S}$ 计算 $\vec{T}' := \vec{U}_i \oplus \vec{S}_i$, 并基于 $T'[g_i] = 1$ 的特性提取 g_i.

• $\mathcal{S}$ 计算 $k_1 := r_{\mathsf{ind}} + y_1 - g_1$.

• $\mathcal{S}$ 计算 $k_i := k_1 + g_1 - g_i + y_i - y_1 - r_{\mathsf{ind}} + k_i'$, $i \in [2, m]$.

• $\mathcal{S}$ 将 k_i (对于 $i \in [m]$) 发送给 P_0.

不可区分性. 本文证明理想世界中输入的消息和 P_1 的输出与真实世界中的消息和输出是不可区分的.

断言3　理想世界 $\mathsf{Ideal}_{\mathcal{F}_{\mathsf{batch\text{-}eq}}, \mathcal{S}, \mathcal{Z}}(1^\lambda)$ 与真实世界 $\mathsf{Real}^{\mathcal{F}_{\mathsf{RVOSE}}}_{\Pi_{\mathsf{batch\text{-}eq}}, \mathcal{A}, \mathcal{Z}}(1^\lambda)$ 是完美不可区分的.

证明　$\mathsf{Ideal}_{\mathcal{F}_{\mathsf{batch\text{-}eq}}, \mathcal{S}, \mathcal{Z}}(1^\lambda)$ 和 $\mathsf{Real}^{\mathcal{F}_{\mathsf{RVOSE}}}_{\Pi_{\mathsf{batch\text{-}eq}}, \mathcal{A}, \mathcal{Z}}(1^\lambda)$ 之间有两个输入输出不同的地方.

• $\mathcal{F}_{\mathsf{RVOSE}}$ 对 P_1 的输出在理想世界和真实世界之间是不可区分的.

• 在理想世界中, $k_1 := r_{\mathsf{ind}} - y_1 - g_1$, 其中 r_{ind} 和 g_1 是均匀随机的. 对于 $i \in [2, m]$, $k_i := k_1 + g_1 - g_i + y_i - y_1 - r_{\mathsf{ind}} + k_i'$, 其中 g_1 和 g_i 是均匀随机的. 在真实世界中, $k_i := x_i + \varepsilon_i$, 其中 ε_i 是均匀随机的. 因此, k_i 在理想世界和真实世界中是不可区分的.

6　电路隐私集合求交方案及其应用

本文首先基于不经意多项式评估 (第 4 节) 和批量相等检测 (第 5 节) 构造了电路隐私集合求交方案, 其与图 1 所提框架相对应. 此外, 本文基于电路隐私集合求交方案描述了常用基于交集数据的对称函数计算方案的实例化过程, 包括基数计算、交集数据关联值求和.

6.1　电路隐私集合求交

电路隐私集合求交的实现思路如图 1 所示, 其分为数据集本地分类、分类数据不经意编码及交集检测与共享三部分. 本小节基于此思路将电路隐私集合求交的具体构造在算法 6 中进行描述.

算法 6 电路隐私集合求交协议.

参数: 哈希函数 $h_1, h_2, h_3 : \{0,1\}^* \to [\beta]$, 布谷鸟哈希表的大小 $\beta := (1+\epsilon)$, 扩展因子 ϵ, 数据集 X 和 Y 的大小 m, 数据长度为 n, 朴素哈希表的每个桶的数据个数 μ;

输入: P_0 输入数据集 X, P_1 输入数据集 Y, $|X| = |Y| = m$.

协议: (1) 本地数据集分类.

- P_0 基于布谷鸟哈希算法将数据集 X 进行映射和存储, 生成布谷鸟哈希表 T_0, 其中 $|T_0| = \beta, |T_0[i]| = 1$ $(i \in [\beta])$;
- P_1 基于朴素哈希算法将数据集 Y 进行映射和存储, 生成朴素哈希表 T_1, 其中 $|T_1| = \beta, |T_1[i]| = \mu$ $(i \in [\beta])$. 如果 $T_1[i]$ 中的数据个数小于 μ, 则采样随机数补齐;

(2) 分类数据不经意编码.

- 对于 $i \in [\beta]$, P_1 随机生成 r_i, 并基于 $T_1[i]$ 中存储的数据本地计算多项式 $f_i(y) = (y - y_1)\cdots(y_\mu) + r_i = a_\mu y^\mu + \cdots + a_1 y + a_0$;
- 对于 $i \in [\beta]$, P_0 和 P_1 调用 $\mathcal{F}^{\mu}_{\mathsf{OPE}}[x_i, f_i(y)]$: P_0 输入 $T_0[i]$ 中存储的数据 x_i, P_1 输入 $f_i(y)$. 协议结束, P_0 获得随机数 s_i. 当且仅当 $x_i \in Y_i$, $s_i = r_i$, 其中 Y_i 为 $T_1[i]$ 中存储的数据;

(3) 交集检测与共享.

- P_0 和 P_1 调用 $\mathcal{F}^{\beta,n}_{\mathsf{batch\text{-}eq}}[R, S]$: P_0 输入 $S := \{s_1, \ldots, s_\beta\}$, P_1 输入 $R := \{r_1, \ldots, r_\beta\}$. 协议结束, P_0 收到 $[\vec{E}]_0$, P_1 收到 $[\vec{E}]_1$, 其中对于 $i \in [\beta]$, $E_i = [E_i]_0 \oplus [E_i]_1 = \mathbf{1}\{s_i = r_i\}$.

6.2　基于交集数据的对称函数计算

电路隐私集合求交方案使得交集数据在参与方之间共享. 因此, 参与双方可以在此基础上基于通用两方安全计算电路实现基于交集数据的任意对称函数计算. 然而, 针对具体的对称函数, 可以基于不经意传输协议构造相较于通用安全计算电路更高效的实例化方式. 因此, 本文基于交集数据的基数、交集数据关联值求和计算的实例化思路分别在图 2 和 3 中进行展示.

在隐私集合交集数据的个数统计中, P_0 和 P_1 分别输入数据集 X 和 Y, 协议执行后, P_1 仅获得两个数据集的交集个数, 而不泄漏具体的交集数据. 在基于电路隐私集合求交方案实例化的过程中, 通过统计 $\vec{E}$ 的汉明 (Hamming) 重量评估交集数据的个数. 具体地, P_0 将 $[\vec{E}]_0$ 直接发送给 P_1, 则 $\sum_{i=1}^{\beta}[E_i]_0 \oplus [E_i]_1$ 为隐私集合交集个数.

在隐私集合交集数据关联值求和中, P_0 输入数据集 X, P_1 输入 (Y, V), 对于每一个隐私数据 y_i, 存在一个与其对应的关联值 v_i. 协议执行后, P_1 获得所有交集数据 y_i 对应的 v_i 的和. 在隐私集合交集数据关联值求和的实例化过程中, 参与双方 P_0 和 P_1 在执行电路隐私集合求交之后, 对于 $i \in [\beta]$, P_1 均匀采样随机数 r_i, 并本地计算 $R := \sum_{i=1}^{\beta} r_i$. 然后, P_0 和 P_1 调用二选一的不经意传输协议. 具体地, P_1 通过 $[E_i]_1$ 设置输入 (m_0, m_1). 如果 $[E_i]_1 = 0$, 则 $m_0 = r_i$, $m_1 = r_i + v_i$; 如果 $[E_i]_1 = 1$, 则 $m_0 = r_i + v_i$, $m_1 = r_i$. P_0 将 $[E_i]_0$ 作为选择比特获得数据 s_i. 显然, 如果 $[E_i]_0 \oplus [E_i]_1 = 0$, 则 $s_i = r_i$; 如果 $[E_i]_0 \oplus [E_i]_1 = 1$, 则 $s_i = r_i + v_i$. 因此, $s_i = r_i + E_i v_i$. 基于此结论, P_0 本地计算 $S := \sum_{i=1}^{\beta} s_i$, 并将其发给 P_1, 则 $S - R = \sum_{i=1}^{\beta} E_i v_i$ 即为交集数据关联值的和.

7　性能评估与分析

本文分别对所提的不经意多项式评估方案 (第 4 节) 与批量相等检测 (第 5 节) 进行了实验评估, 并将不经意多项式评估方案的性能与 PSTY19 [24], RS21 [17] 和 CGS22 [16] 中的不经意编码方案进行了对比. 此外, 本文将所提的电路隐私集合求交方案从数据集分类、不经意编码和交集检测与共享三部分与 PSTY19 [24], RS21 [17] 和 CGS22 [16] 进行了比较.

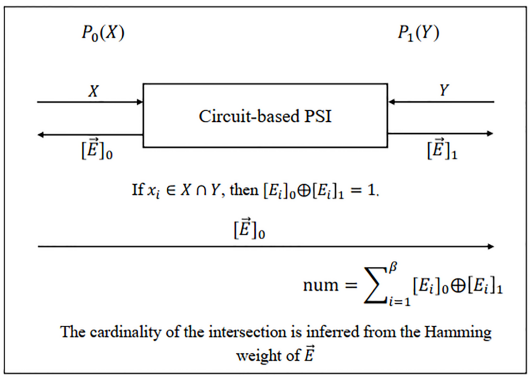

图 2　隐私集合交集基数统计协议构建图.

Figure 2　The construction of PSI-Cardinality.

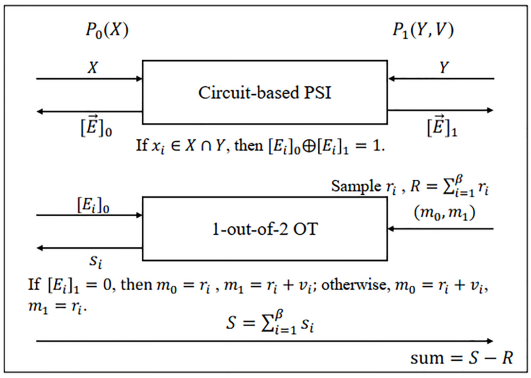

图 3　隐私集合交集数据关联值求和协议构建图.

Figure 3　The construction of PSI-Sum of associated values.

7.1　实验设置

实验环境. 本文的实验基于 C++ 实现, 实验的计算机硬件环境为 PC (CPU: Intel(R) Core(TM) i5-7500 CPU @ 40 GHz, RAM: 16.0 GB, OS: Ubuntu 18.04.2), 本文使用 Linux 的 t_c 命令模拟网络环境. 本文的局域网 (local-area network, LAN) 设置为 20 Gbps 的网络带宽和 0.01 ms 的往返时延 (round-trip time, RTT), 城域网 (metropolitan-area network, MAN) 设置为 400 Mbps 的带宽和 20 ms 的 RTT, 广域网 (wide-area network, WAN) 设置为 10 Mbps 的带宽和 100 ms 的 RTT.

实验参数. 本文的计算安全参数 $\lambda = 128$, 统计安全参数 $\kappa = 40$. 本文使用开源的 OT 库[1)]实例化了 $\mathcal{F}_{\mathsf{VOLE}}$, 并使用开源代码[2)]实例化了 $\mathcal{F}_{\mathsf{RVOSE}}$. 此外, 本文基于源代码[3)4)5)]对方案 PSTY19 [24], RS21 [17] 和 CGS22 [16] 进行了实现. 在与上述工作进行实验对比时, 本文设置数据集大小 $n \in \{2^{10}, 2^{11}, \ldots, 2^{17}, 2^{18}\}$.

7.2　实验评估

本文从运行时间和通信开销两方面对不经意编码和电路隐私集合求交方案的性能进行了评估, 并与相关工作进行比较.

7.2.1　不经意编码

目前存在基于不同原语的不经意编码方案, PSTY19 [24] 的不经意编码方案基于 PNI 和 OPRF 构造, RS21 [17] 基于 OKVS 和 OPRF 实现, CGS22 [16] 通过 RB-OPPRF 实现. 本文提出了基于 VOLE 的 OPE 实现不经意编码. 因此, 本文通过实验对上述基于不同原语的不经意编码方案进行评估和比较.

运行时间. 在不同网络环境和数据集大小下, 基于不同密码原语的不经意编码方案的运行时间评估结果如表 2 所示. 其中, PNI 和 RB-OPPRF 因涉及高维方程组求解, 运行时间较长, 而 OKVS 通过构造稀疏多项式对方案的运行时间进行优化, 本文的方案通过多项式分解使得 OPE 基于 VOLE 快速实现, 具有最优的运行时间. 例如, 在 WAN 下, 对于 2^{18} 的数据集大小, 本文的方案比 PNI 快 6.2 倍, 比 OKVS 快 1.3 倍, 比 RB-OPPRF 快 2.7 倍. 本文所提不经意编码方案在不同网络环境下均具有明显的优势. 此外, 表 1 描述了基于不同原语的不经意编码协议的通信轮数. 相较于 PNI 与 OKVS 的实现, 本文的方案在通信轮数方面具有明显优势. 因此, 随着带宽的减少和时延的增加, 本文的方案在运行时间方面的优势愈加明显.

通信开销. 在不同数据集大小下, 不经意编码方案的通信开销如表 3 所示. 其中, PNI 的通信开销较低, 因

1) https://github.com/osu-crypto/libOTe.
2) https://zenodo.org/records/14580231.
3) https://github.com/encryptogroup/OPPRF-PSI.
4) https://github.com/Visa-Research/volepsi.
5) https://aka.ms/2PC-Circuit-PSI.

表 2　基于不同原语的不经意编码方案在 n 个编码数据下的运行时间 (ms).

Table 2　Running time (ms) of the oblivious encoding based on different primitives for n encoded elements.

n	LAN				MAN				WAN			
	PNI	OKVS	RB-OPPRF	Ours	PNI	OKVS	RB-OPPRF	Ours	PNI	OKVS	RB-OPPRF	Ours
2^{10}	345.18	14.2	18.81	12.78	386.56	145.31	210.25	120.76	2216.57	271.50	904.117	237.21
2^{11}	407.55	23.96	29.83	20.11	576.70	258.42	354.49	182.84	2596.42	349.35	1100.51	330.69
2^{12}	457.06	30.45	38.59	28.87	709.69	315.36	435.17	256.71	2739.44	656.42	1350.22	537.41
2^{13}	507.81	59.77	71.42	56.98	990.44	446.82	571.66	355.51	3635.98	1112.54	1601.48	733.79
2^{14}	649.87	110.15	122.51	101.20	1180.31	524.12	748.78	466.15	4552.53	1501.45	2201.24	1239.22
2^{15}	919.83	222.63	251.29	206.71	1484.62	674.75	917.37	562.78	7271.84	2180.15	3448.60	1862.29
2^{16}	1905.58	551.85	603.12	520.75	2571.33	1131.45	1535.59	1093.79	11552	3457.2	5733.5	3175.86
2^{17}	3711.36	951.82	1148.18	916.38	4463.94	2116.35	2056.32	1718.31	23943	6921.52	10138	5827.98
2^{18}	5036.46	1909.56	2363.35	1812.02	6871.54	3309.82	4053.56	2891.72	44463	9675.4	19179	7179.91

表 3　基于不同原语的不经意编码方案在 n 个编码数据下的通信开销 (MB).

Table 3　Communication cost (MB) of the oblivious encoding based on different primitives for n encoded elements.

n	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}
PNI	0.29	0.59	1.19	2.39	4.79	9.59	19.79	38.39	76.79
OKVS	0.79	0.90	1.90	3.56	6.82	13.46	25.78	49.63	85.79
RB-OPPRF	0.23	0.38	1.05	2.04	4.03	8.02	15.93	31.81	63.56
Ours	0.16	0.27	0.38	0.81	1.74	3.37	7.94	16.83	35.56

为它仅涉及插值多项式系数的发送和接收. 相比之下, OKVS 在稀疏多项式生成过程中因为需要构造一个扩展矩阵, 因此引入了额外的通信开销. RB-OPPRF 优化了批量 OPPRF 构造, 在批处理时实现了较低的通信开销. 本文的方案利用 VOLE 实现了较低的均摊通信开销, 其优势对于大数据集更为明显, 且数据集规模越大, 本文方案的优势越明显.

7.2.2　电路隐私集合求交

在此部分, 本文将所提电路隐私集合求交协议与 PSTY19 [24], RS21 [17] 和 CGS22 [16] 在运行时间和通信开销方面进行了对比. 基于本文提出的框架, 本文将每个协议的开销分为三部分: 数据集分类、不经意编码和交集检测与共享. 本文对每个部分均进行了实验评估, 并对每部分占总方案开销的比例进行了分析.

运行时间. 在不同网络环境和数据集大小下, 电路隐私集合求交的运行时间如表 4 所示. 总体地, 数据集分类由各参与方本地执行, 因此其运行时间不受网络环境的影响. 此外, RS21 [17], CGS22 [16] 和本文的方案使用相同的数据集分类方法, 因此不同方案此部分的运行时间相同. 在不同的网络环境下, 不经意编码与交集共享决定协议运行时间的主要部分, 占总运行时间的 90% 以上, 且随着数据集的增大, 交集共享所占的运行时间比例增加. 此外, 在不同网络环境和数据集大小下, 本文的方案在不经意编码和交集共享方面均优于其他方案.

通信开销. 电路隐私集合求交方案在不同数据集大小下的通信开销如表 5 所示. 由于数据集分类由各参与方本地完成, 因此不产生通信开销. 对于整体方案, 本文的 CPSI 在不同数据集大小下, 相比于其他方案在通信开销方面具有明显的优势. 此外, CPSI 的通信开销由两部分组成: 不经意编码和交集数据共享. 不经意编码阶段的核心原语为 VOLE, 相较于批量相等检测, 其通信开销随着数据量的增加增长缓慢. 因此, 随着数据量大增加, 交集共享部分的通信开销占比逐渐增加.

表 4　CPSI 的运行时间 (s) 与 PSTY19 [24], RS21 [17] 和 CGS22 [16] 的对比, 其中 n 为数据集大小.

Table 4　Running time (s) of the CPSI compared with PSTY19 [24], RS21 [17], and CGS22 [16], where n is the dataset size.

	Protocol	PSTY19 [24]				RS21 [17]				CGS22 [16]				Ours			
	n	2^{12}	2^{14}	2^{16}	2^{18}	2^{12}	2^{14}	2^{16}	2^{18}	2^{12}	2^{14}	2^{16}	2^{18}	2^{12}	2^{14}	2^{16}	2^{18}
LAN	Dataset classification	0.01	0.05	0.19	0.65	0.005	0.03	0.12	0.54	0.005	0.03	0.12	0.54	0.005	0.03	0.12	0.54
	Oblivious encoding	0.46	0.65	1.91	5.04	0.03	0.11	0.55	0.91	0.04	0.12	0.60	2.36	0.03	0.10	0.52	1.81
	Membership sharing	0.39	0.54	1.04	4.68	0.39	0.54	1.04	4.68	0.46	0.64	1.52	5.19	0.33	0.51	0.92	3.64
	Total running time	0.86	1.24	3.14	10.37	0.43	0.68	1.71	6.13	0.51	0.79	2.24	6.09	0.37	0.64	1.56	5.99
MAN	Dataset classification	0.01	0.05	0.19	0.65	0.005	0.03	0.12	0.54	0.005	0.03	0.12	0.54	0.005	0.03	0.12	0.54
	Oblivious encoding	0.71	1.18	2.57	6.87	0.31	0.52	1.13	3.31	0.44	0.75	1.54	4.05	0.26	0.47	1.09	2.89
	Membership sharing	0.72	1.06	3.54	7.78	0.72	1.06	3.54	7.78	1.39	1.93	3.63	10.99	0.67	1.14	2.58	5.13
	Total running time	1.44	2.29	6.30	15.30	1.04	1.61	4.79	11.63	1.84	2.71	5.29	15.58	0.94	1.64	3.79	8.56
WAN	Dataset classification	0.01	0.05	0.19	0.65	0.005	0.03	0.12	0.54	0.005	0.03	0.12	0.54	0.005	0.03	0.12	0.54
	Oblivious encoding	2.74	4.55	11.55	44.46	0.66	1.50	3.46	9.68	1.35	2.20	5.73	19.18	0.54	1.24	3.18	7.18
	Membership sharing	2.60	4.54	14.91	126.54	2.60	4.54	14.91	126.54	8.90	17.24	57.62	235.63	2.35	3.93	12.75	112.25
	Total running time	5.35	9.14	26.65	171.65	3.27	6.07	18.49	136.76	10.26	17.47	63.47	254.81	2.90	5.20	16.05	119.97

表 5　CPSI 的通信开销 (MB) 与 PSTY19 [24], RS21 [17] 和 CGS22 [16] 的比较, 其中 n 为数据集大小, "–" 表示此阶段不存在通信开销.

Table 5　Communication cost (MB) of CPSI compared with PSTY19 [24], RS21 [17], and CGS22 [16], where n is the dataset size, and "–" indicates that there is no communication cost in this part.

n	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}
PSTY19 [24]	2.2	4.4	9.2	19.8	40.6	82.4	162.3	325.6	650.2
RS21 [17]	5.1	7.7	13.4	20.6	32.7	50.3	171.6	363.2	686.4
CGS22 [16]	1.6	3.1	6.2	11.9	24.1	48.4	96.9	193.8	387.3
Ours	0.8	1.6	3.0	5.9	12.1	24.1	49.4	99.7	201.4
Communication cost breakdown by step									
Dataset classification	–	–	–	–	–	–	–	–	–
Oblivious encoding	0.2	0.3	0.4	0.8	1.7	3.4	7.9	16.8	35.6
Total running time	0.6	1.3	2.6	5.1	10.4	20.7	41.5	82.9	165.8

8　结论

本文对现有的电路隐私集合求交协议进行了总结和分析, 并将其实现过程分为三部分: 数据集本地分类、分类数据不经意编码及交集成员检测与共享. 通过理论分析与实验评估可知后两部分的运行时间占整体方案的 90% 以上, 通信开销占整体方案的 100%. 基于此结论, 本文首先提出了一种基于 OPE 的不经意编码方案. 不同于依赖计算开销较大的插值多项式的实现方式, 本文的不经意编码方案将高阶多项式计算分解为多个线性多项式计算, 通过用简单的算术运算 (乘法和加法) 替代插值多项式计算的复杂过程, 本文将不经意编码的计算复杂度从 $\mathcal{O}(n^2)$ 降低到 $\mathcal{O}(n)$. 此外, 本文提出了一种通信高效的批量相等检测方法, 从而实现交集的检测和共享. 与传统方案重复独立调用相等检测的思路不同, 本文通过不经意索引旋转策略的设计, 将批量相等检测问题转换为使用单一索引的不经意检索问题, 实现了目前最低的在线阶段通信开销. 实验结果表明, 本文提出的隐私集合求交方案在运行时间方面比 PSTY19 快 2.3×, 比 RS21 快 2.3×, 比 CGS22 快 1.37×. 在通信开销方面, 本文所提协议的通信开销仅是 PSTY19 的 31%, RS21 的 29% 和 CGS22 的 52%. 此外, 通过对数据不经意编码及交集成员检测与共享的计算和通信开销分别占总开销的比例进行分析, 可知交集成员检测与共享的开销占总开销的

50% 以上. 因此, 在未来工作中, 我们将致力于进一步优化批量相等检测的效率, 以提高电路隐私集合求交方案的整体性能.

参考文献

1 Kolesnikov V, Kumaresan R, Rosulek M, et al. Efficient batched oblivious PRF with applications to private set intersection. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016

2 Chase M, Miao P. Private set intersection in the internet setting from lightweight oblivious PRF. In: Proceedings of the 40th Annual International Cryptology Conference, Santa Barbara, 2020. 34–63

3 Applebaum B, Konstantini N. Actively secure arithmetic computation and VOLE with constant computational overhead. In: Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, 2023. 190–219

4 Roy L. Softspokenot: quieter ot extension from small-field silent vole in the minicrypt model. In: Proceedings of the 42nd Annual International Cryptology Conference, Santa Barbara, 2022. 657–687

5 Kales D, Rechberger C, Schneider T, et al. Mobile private contact discovery at scale. In: Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), 2019. 1447–1464

6 Huang C T, Zhang F, Sun X C, et al. A survey of private set intersection technology and finance practice. Inform Commun Tech Policy, 2021, 47: 50 [黄翠婷, 张帆, 孙小超, 等. 隐私集合求交技术的理论与金融实践综述. 信息通信技术与政策, 2021, 47: 50]

7 Lin C, He D B, Zeadally S, et al. SECBCS: a secure and privacy-preserving blockchain-based crowdsourcing system. Sci China Inf Sci, 2020, 63: 130102

8 Chen H X, Huang X Y, Wu W, et al. Efficient and secure image authentication with robustness and versatility. Sci China Inf Sci, 2020, 63: 222301

9 Liu Y, Zhang B, Ma Y, et al. iPrivJoin: an ID-private data join framework for privacy-preserving machine learning. IEEE Trans Inform Forensic Secur, 2023, 18: 4300–4312

10 Tong Y W, Feng Q, Luo M, et al. Multi-party privacy-preserving decision tree training with a privileged party. Sci China Inf Sci, 2024, 67: 182303

11 Ling G, Tang F, Cai C, et al. P^2FRPSI: privacy-preserving feature retrieved private set intersection. IEEE Trans Inform Forensic Secur, 2023, 19: 2201–2216

12 Wu M, Yuen T H. Efficient unbalanced private set intersection cardinality and user-friendly privacy-preserving contact tracing. In: Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), 2023. 283–300

13 Wang C, Ruan O. Efficient private set intersection-sum with cardinality based on differential private load overestimation mechanism. In: Proceedings of the 3rd International Conference on Cryptography, Network Security and Communication Technology, 2024. 382–388

14 Hu J, Zhao Y, Tan B H M, et al. Enabling threshold functionality for private set intersection protocols in cloud computing. IEEE Trans Inform Forensics Secur, 2024, 19: 6184–6196

15 Branco P, Döttling N, Pu S. Multiparty cardinality testing for threshold private intersection. In: Proceedings of the IACR International Conference on Public-Key Cryptography, 2021. 32–60

16 Chandran N, Gupta D, Shah A. Circuit-PSI with linear complexity via relaxed batch OPPRF. In: Proceedings on Privacy Enhancing Technologies, 2022

17 Rindal P, Schoppmann P. VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE. In: Proceedings of the 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, 2021. 901–930

18 Papakyriakou D, Barbounakis I S. Data mining methods: a review. Int J Comput App, 2022, 183: 5–19

19 Wang C Y, Wang D, Xu G A, et al. Efficient privacy-preserving user authentication scheme with forward secrecy for Industry 4.0. Sci China Inf Sci, 2022, 65: 112301

20 Kang X, Ma Z, Liu Y, et al. Multi-party private edge computing for collaborative quantitative exposure detection of endemic diseases. IEEE Trans Mobile Comput, 2024, 23: 12020–12034

21 Zhong S, Huang X Y. Special focus on security and privacy in blockchain-based applications. Sci China Inf Sci, 2020, 63: 130100

22 Fournier-Viger P, Wang Y, Yang P, et al. TSPIN: mining top-k stable periodic patterns. Appl Intell, 2022, 52: 6917–6938

23 Pinkas B, Schneider T, Zohner M. Scalable private set intersection based on OT extension. ACM Trans Priv Secur, 2018, 21: 1–35

24 Pinkas B, Schneider T, Tkachenko O, et al. Efficient circuit-based psi with linear communication. In: Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, 2019. 122–153

25 Lu T, Kang X, Zhang B, et al. Efficient 2PC for constant round secure equality testing and comparison. Cryptology ePrint

Archive. 2024. https://eprint.iacr.org/2024/949
26 Yang Y, Yang Y, Chen X, et al. DMPSI: efficient scalable delegated multiparty PSI and PSI-CA with oblivious PRF. IEEE Trans Services Comput, 2024, 17: 497–508
27 Miao P, Patel S, Raykova M, et al. Two-sided malicious security for private intersection-sum with cardinality. In: Proceedings of the 40th Annual International Cryptology Conference, Santa Barbara, 2020. 3–33
28 Huang Y, Evans D, Katz J. Private set intersection: are garbled circuits better than custom protocols? In: Proceedings of Network and Distributed System Security (NDSS) Symposium, 2012
29 Chen R M, Chen J R, Huang X, et al. RCCA-SM9: securing SM9 on corrupted machines. Sci China Inf Sci, 2024, 67: 212103
30 Liu X H, Huang X Y, Cheng Z H, et al. Fault-tolerant identity-based encryption from SM9. Sci China Inf Sci, 2024, 67: 122101
31 Naor M, Pinkas B. Oblivious transfer and polynomial evaluation. In: Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing, 1999. 245–254
32 Chen H, Laine K, Rindal P. Fast private set intersection from homomorphic encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017. 1243–1255
33 Hu J, Chen J, Dai W, et al. Fully homomorphic encryption-based protocols for enhanced private set intersection functionalities. Cryptology ePrint Archive, 2023. https://eprint.iacr.org/2023/1407
34 Tueno A, Kerschbaum F. Efficient secure computation of order-preserving encryption. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, 2020. 193–207
35 Cianciullo L, Ghodosi H. Efficient information theoretic multi-party computation from oblivious linear evaluation. In: Proceedings of the IFIP International Conference on Information Security Theory and Practice, 2018. 78–90
36 Ge C, Susilo W, Liu Z, et al. Secure keyword search and data sharing mechanism for cloud computing. IEEE Trans Depend Secure Comput, 2020, 18: 2787–2800
37 Canetti R. Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, 2001. 136–145
38 Boyle E, Couteau G, Gilboa N, et al. Compressing vector ole. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018. 896–912
39 Couteau G. New protocols for secure equality test and comparison. In: Proceedings of the International Conference on Applied Cryptography and Network Security, 2018. 303–320

Privacy set intersection from oblivious polynomial evaluation

Xin KANG[1], Yong ZENG[1*], Zhuzhu WANG[2*], Zhuoran MA[1], Tanren LIU[1], Junjie ZHOU[1], Zhihong LIU[1], Zhuo MA[1] & Jianfeng MA[1]

1. *School of Cyber Engineering, Xidian University, Xi'an 710071, China*
2. *School of Information Science & Technology, Northwestern University, Xi'an 710068, China*
* Corresponding author. E-mail: yzeng@mail.xidian.edu.cn, zzwang@nwu.edu.cn

Abstract Traditional private set intersection (PSI) protocols allow parties to directly compute the intersection of the private sets without revealing their individual set contents. However, they do not support extension computations based on the intersection data. The circuit-based PSI protocol enables parties to perform extension computations on the intersection without leaking the actual intersection elements, thereby constructing a general-purpose privacy-preserving set intersection scheme. In this paper, we first survey and analyze existing work, summarizing a unified framework for circuit-PSI, which consists of three key components: local dataset classification, oblivious encoding of classified data, and intersection membership detection and sharing. Notably, the latter two components account for over 90% of the total computational cost and 100% of the communication cost. Therefore, we propose an oblivious encoding scheme based on oblivious polynomial evaluation for symmetric datasets. By replacing computationally expensive interpolation polynomial systems with efficient first-degree polynomial evaluation, we reduce the computational complexity of oblivious encoding from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. Furthermore, we design a communication-efficient batch equality test protocol for intersection membership detection and sharing. We reformulate the batch equality test as an oblivious retrieval problem and introduce an oblivious index rotation strategy, allowing batch equality test results to be retrieved using a single index. This approach achieves the lowest online communication cost among state-of-the-art protocols. Leveraging these optimizations, our experimental results demonstrate that our private data matching for the compute protocol achieves a speedup of 2.3× over PSTY19 (EUROCRYPT 19), 1.16× over RS21 (EUROCRYPT 21), and 1.37× over CGS22 (PETs 22). Regarding communication efficiency, our protocol requires only 31% of the communication cost of PSTY19, 29% of RS21, and 52% of CGS22.

Keywords private set intersection, circuit-based PSI, oblivious polynomial evaluation, batch equality test