软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(4):1819−1837 [doi: 10.13328/j.cnki.jos.007434] [CSTR: 32375.14.jos.007434]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 兼顾通信轮数与计算开销的门限多方隐私集合交集协议*

张 恩[1,2], 黄昱晨[1], 郑 东[3], 禹 勇[4]

[1](河南师范大学 计算机与信息工程学院, 河南 新乡 453007)

[2](河南省教育人工智能与个性化学习重点实验室, 河南 新乡 453007)

[3](无线网络安全技术国家工程实验室 (西安邮电大学), 陕西 西安 710121)

[4](陕西师范大学 计算机科学学院, 陕西 西安 710119)

通信作者: 张恩, E-mail: zhangenzdrj@163.com

**摘 要**: $(t, N)$ 门限多方隐私集合交集协议 (threshold multi-party private set intersection, TMP-PSI) 允许当指定参与方的集合元素 $x$ 在其余不少于 $t$–1 ($t<N$) 个参与方的私有集合中出现时, 数据元素 $x$ 作为交集结果输出, 在提案投票、金融交易威胁识别、安全评估等场景具有广泛应用. 现有的门限多方隐私集合交集协议运行效率低、通信轮数多且只能由某一个指定参与方获取交集. 针对这些问题, 设计一种基于弹性秘密共享的参与方门限测试方法, 结合不经意键值对存储 (oblivious key-value store, OKVS) 提出一种 TMP-PSI 方案, 能够有效减少计算开销和通信轮数. 为了满足多参与方获取私有集合中交集信息的需求, 提出第 2 种拓展门限多方隐私集合交集 (extended threshold multi-party private set intersection, ETMP-PSI) 协议对份额分发方式进行改变, 与第 1 种方案相比, 秘密分发者和秘密重构方没有额外增加通信轮数和计算复杂度, 实现了多参与方获取私有集合中的交集元素. 所设计的协议在数据集合大小为 $n=2^{16}$ 的三方场景下运行时间为 6.4 s (TMP-PSI) 和 8.7 s (ETMP-PSI), 与现有的门限多方隐私集合交集协议相比, 重构方和分发方的通信复杂度由 $O(nNt\log n\lambda)$ 降为 $O(bN\lambda)$.

**关键词**: 门限多方隐私集合交集协议; 通信轮数; 计算开销; 弹性秘密共享; 不经意键值对存储

**中图法分类号**: TP309

中文引用格式: 张恩,黄昱晨,郑东,禹勇.兼顾通信轮数与计算开销的门限多方隐私集合交集协议.软件学报,2026,37(4):1819–1837. http://www.jos.org.cn/1000-9825/7434.htm

英文引用格式: Zhang E, Huang YC, Zheng D, Yu Y. Threshold Multi-party Private Set Intersection Protocol Balancing Communication Rounds and Computational Overhead. Ruan Jian Xue Bao/Journal of Software, 2026, 37(4): 1819–1837 (in Chinese). http://www.jos.org.cn/1000-9825/7434.htm

## Threshold Multi-party Private Set Intersection Protocol Balancing Communication Rounds and Computational Overhead

ZHANG En[1,2], HUANG Yu-Chen[1], ZHENG Dong[3], YU Yong[4]

[1](School of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China)

[2](Key Laboratory of Artificial Intelligence and Personalized Learning in Education of Henan Province, Xinxiang 453007, China)

[3](National Engineering Laboratory of Wireless Network Security Technology (Xi'an University of Posts and Telecommunications), Xi'an 710121, China)

[4](School of Computer Science, Shaanxi Normal University, Xi'an 710119, China)

**Abstract**: The $(t, N)$ threshold multi-party private set intersection (TMP-PSI) protocol allows a given party's data element $x$ to appear in the private sets of no fewer than $t$–1 other parties. The data element $x$ is then output as the intersection result, which is widely applied in scenarios such as proposal voting, financial transaction threat identification, and security assessment. Existing threshold multi-party private set intersection protocols suffer from low efficiency, high communication rounds, and a limitation that only a specific participant can

* 基金项目: 国家自然科学基金 (62372157)
收稿时间: 2024-08-02; 修改时间: 2024-11-22; 采用时间: 2025-03-17; jos 在线出版时间: 2025-07-30
CNKI 网络首发时间: 2025-07-31

obtain the intersection. To address these issues, this study proposes a threshold testing method based on robust secret sharing (RSS) and a TMP-PSI scheme combined with oblivious key-value store (OKVS), which effectively reduces both computational overhead and the number of communication rounds. To meet the demand for multiple participants to access the intersection information from their private sets, this study also proposes a second extended threshold multi-party private set intersection (ETMP-PSI) protocol, which modifies the share distribution method. Compared to the first scheme, the secret distributor and secret reconstructor do not incur additional communication rounds or computational complexity, allowing multiple participants to obtain the intersection elements from their private sets. The proposed protocol runs in 6.4 seconds (TMP-PSI) and 8.7 seconds (ETMP-PSI) in a three-party scenario with a dataset size of $n=2^{16}$. Compared to existing threshold multi-party private set intersection protocols, the communication complexity between the reconstructor and distributor is reduced from $O(nNt\log n\lambda)$ to $O(bN\lambda)$.

**Key words**: threshold multi-party private set intersection (TMP-PSI) protocol; communication round; computational overhead; robust secret sharing; oblivious key-value store (OKVS)

数字技术作为世界科技革命和产业变革的先机, 日益融入经济社会发展各领域全过程, 数据资源已经成为数字经济时代的基础性生产要素. 为了保护用户隐私信息在数据交互过程中的安全, 研究人员提出了各种隐私保护方法, 其中安全多方计算是保护隐私的有效手段之一. 隐私集合交集 (private set intersection, PSI)[1−14]作为安全多方计算的一个重要应用, 与其他隐私保护框架[15,16]被广泛应用于医疗、金融和社交网络等领域.

隐私集合交集协议允许各个参与方之间执行协议获得所有参与方持有数据元素的交集, 但不泄露除了交集以外的任何信息. 隐私集合交集技术在数据交互的过程中, 发挥了保护个人隐私的关键作用, 允许不同实体在共享数据时保护数据的隐私, 同时允许参与者在不暴露私有信息的情况下进行集合运算.

但是标准隐私集合交集协议在现实生活中的某些场景并不适用, 例如议案投票时, 投票支持人数超过一定数量即可, 并不需要全部投票人员都支持才通过议案. 此外可用于金融机构协作识别交易威胁: 某个金融机构 A 需要检测用户账户是否存在交易威胁且不泄露金融机构的隐私信息, 假设采用多方隐私集合交集协议 (multi-party private set intersection, MP-PSI)[17−25], 将金融机构 A 中的异常用户与其他金融中心进行联合筛查, 一旦金融机构 A 中的异常用户在其他金融机构都存在异常交易行为, 就标记为交易威胁, 并上传至监管名单, 但该交易威胁的设置过于严格, 因为用户不一定在全部金融机构都存在异常交易行为. 此场景可以采用门限多方隐私集合交集协议判断交易威胁, 设置预警阈值 $t$, 在检测时间段内, 金融中心 A 的异常交易用户, 在其余 $t-1$ 个金融中心中存在异常交易行为, 将被标记存在交易威胁. 此协议同样适用于网络操作中心协作识别常见威胁, 只要 $t$ 个网络中心受到攻击, 就被设置为常见威胁[26].

针对问题, Bay 等人[17]提出了门限多方隐私集合交集协议 (TMP-PSI) 的概念, 允许一个服务器端 $P_1$ 和多个客户端 $P_2,\ldots,P_N$, 输入他们的私有数据集 $X_1,\ldots,X_N$, $N$ 个参与方共同计算服务器端数据集 $X_1$ 中, 哪些数据元素是至少 $t$ ($t<N$) 个参与方共同拥有的, 并且除了服务器端以外, 其他参与方不知道任何信息, 如图 1 所示.

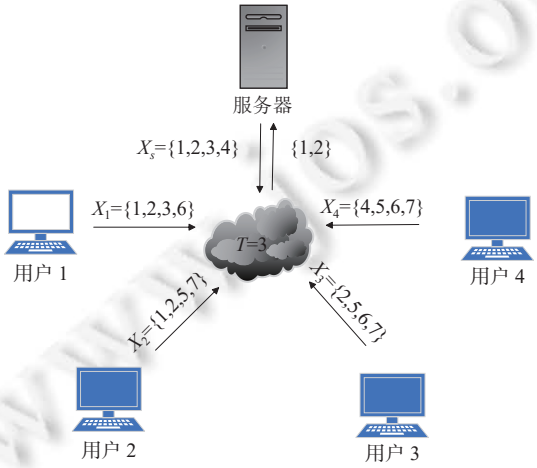



图 1 门限多方 PSI 示意图

## 1 相关工作

现有门限隐私集合的研究主要分为两种, 一种是数据集门限 (multi-party threshold private set intersection, MP-TPSI)[27−32], 各个参与方私有集合中, 数据元素的交集数量达到门限值. 张恩等人[27]采用混淆布隆过滤器和秘密共享的方式实现了数据集门限测试方法, 设计了一种多方门限隐私集合交集协议. Liu 等人[28]将确定型门限通过 GS 协议转化成概率型门限, 设计了一种概率门限测试协议, 使得协议性能显著提高. Mohanty 等人[29]提出了一种两方抗量子的 TPSI 协议, 通过可信第三方生成抗量子密钥来完成两方之间的隐私集合交集操作, 文中仅提供了理论基础, 并未实现. Ghosh 等人[30]提出了一种 TPSI 协议, 基于加性同态加密设计门限测试方法, 使得协议的通信复杂度仅依赖于阈值 $t$. Badrinarayanan 等人[31]将 Ghosh 等人[30]的工作由两方拓展到多方, 研究了多方设置下 ($N>2$) 门限测试的通信复杂度. Ghosh 等人[32]将 TPSI 协议分为隐私集合交集基数测试 (private intersection cardinality test, PICT) 和 PSI 两个部分, 通过通用安全计算框架实现 PICT 功能, 实现多方设置下更好的通信复杂度.

另一种是参与方数量门限 (TMP-PSI)[17,24,26,33], 拥有相同数据元素的参与方数量达到门限值, 则该数据元素作为交集输出给指定方. Bay 等人[17]使用同态加密和加密布隆过滤器结合的方式设计了一种判断参与方门限的方法. 这个协议的性能瓶颈在于同态加密和加密布隆过滤器设计的门限测试, 在交互过程中的通信开销随着数据集的增多呈指数型增长, 使得协议的运行效率受到较大影响. Chandran 等人[24]提出了一个 Quorum PSI 协议, 他们采用了电路的方式实现了此功能, 选择一个参与方 $P_1$ 作为领导者, $P_1$ 拥有的某个数据元素 $x$, 如果在不少于 $t$ 个参与方的集合中存在, 则作为交集输出给 $P_1$, 此方案仅有领导者 $P_1$ 得到输出, 其他参与方不得到任何输出. Chandran 等人使用通用电路框架构造协议, 因此随着数据集和参与方数量的增多, 需要生成大量的三元组以及更多的通信轮数, 且无法适应门限值的变化. Mahdavi 等人[26]采用多次执行多方 PSI 技术实现超阈值隐私集合交集协议, 每次 PSI 协议仅有一个参与方可以得到自己集合中满足条件的元素, 想要所有参与方都得到输出需要执行 $N$ ($N$ 为参与方数量) 次多方 PSI 协议, 随着参与方数量的增多, 执行多方 PSI 次数过多, 无法实现有效拓展. 魏立斐等人[33]提出了一个半诚实模型下高效的双云辅助超阈值隐私集合交集协议, 此协议额外引入第三方云的强大算力, 将复杂的重构操作交给第三方云来辅助计算. 在重构阶段计算复杂度为 $O(n(N\log n/t)^{2t})$, 随着门限值 $t$ 的增加, 协议重构阶段计算开销增长过快, 导致协议整体运行效率下降.

本文构造了两种门限多方隐私集合交集协议, 第 1 种协议仅有一个指定参与方获取自己数据集合中达到门限的交集信息. 第 2 种拓展门限多方隐私集合交集协议, 实现了多参与方获取私有集合中的交集元素, 除此以外得不到其他任何信息, 保证了协议的安全性, 如图 2 所示.

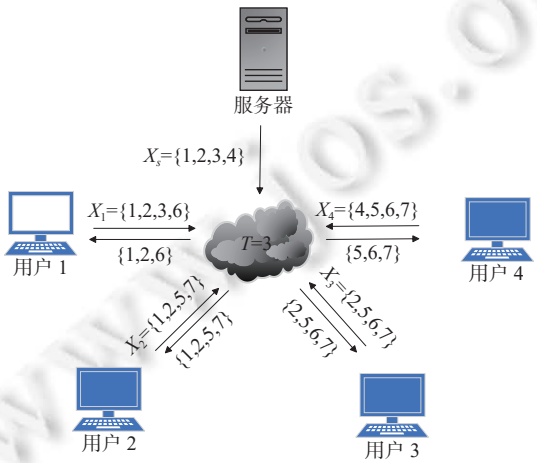



图 2 拓展门限多方 PSI 示意图

本文的主要贡献如下.

(1) 提出了一种兼顾通信轮数与计算开销的参与方门限测试方法, 设计的门限测试方案采用弹性秘密共享和不经意键值对存储, 实现了任意门限下的高效计算. 秘密分发者在本地生成所需的秘密份额, 仅需要一轮通信就可以完成所有秘密份额的发送, 参与方通过秘密份额隐藏自己的私有数据集合后执行重构操作, 不仅保护了交集以外其他信息, 而且能够有效减少通信轮数和计算开销.

(2) 针对现有的 TMP-PSI 协议只能单参与方获取交集, 无法满足多参与方获取私有集合中交集信息的需求, 本文通过对弹性秘密共享份额分发阶段进行拓展, 实现了多参与方获得私有集合中达到门限值的交集信息, 相比较于第 1 种协议不需要额外增加通信轮数. 本文设计的 ETMP-PSI 协议与 Mahdavi 等人[26]和魏立斐等人[33]提出的超阈值隐私集合交集协议相比, 重构方计算复杂度由 $O(nNt\log n\lambda)$ 降为 $O(bN\lambda)$ ($n$ 是数据集合大小, $N$ 是参与方数量, $t$ 是门限值, $\lambda$ 是计算安全参数).

(3) 通过实验测试了两种协议的性能, 从计算和通信复杂度方面、不同数据集大小和不同参与方数量等多种情况下给出了详细的分析和实验数据, 实验结果表明两种方案与现有的方案相比拥有更好的性能.

## 2 基础知识

本节介绍本文协议中用到的相关技术以及安全模型.

### 2.1 布谷鸟哈希和朴素哈希表

布谷鸟哈希 (cuckoo hash) 技术最早于 2001 年由 Pagh 等人[34]提出的一种哈希技术, 不仅可以高效地构造哈希表, 而且数据元素的插入算法性能较高. 布谷鸟哈希表是通过多个 (通常是 2、3 或 4 个) 不同的哈希函数实现的, 其特性是哈希表中的每个位置 (*bin*) 只存储一个数据元素, 具体流程如下所示.

(1) 初始化一个长度为 $b$ 的哈希表 $T$, 选取 $k$ 个 $hash:\{0,1\}^*\to\{0,1\}^\lambda$ 函数 $H_1,\ldots,H_k$.

(2) 使用某个哈希函数 $H_i(\cdot)$, $i\in[1,k]$ 将数据元素 $x$ 映射到哈希表 $T$ 的 $H_i(x)$ 位置, 每个 *bin* 中只允许存储一个数据元素. 如果此位置不为空则踢出此位置原有的数据元素, 将其放入待插入队列中重新插入.

(3) 当上述替换操作达到一定次数后, 则将未插入的数据元素放置在额外的存储空间 (*stash*) 中.

(4) 对于表中的空位置使用随机值填充.

本文采用无 *stash* 的布谷鸟哈希表, 如图 3 所示. 在布谷鸟哈希中, 有多个哈希函数 (图 3 中使用两个哈希函数 $H_1$ 和 $H_2$) 和数据元素 (图 3 中的 $x_1$、$x_2$、$x_3$、$x_4$).

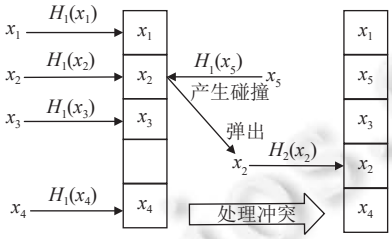



图 3 布谷鸟哈希示意图

初始映射: 首先使用哈希函数将数据元素映射到哈希表中的位置, $x_1$ 使用 $H_1$ 映射到位置 $H_1(x_1)$, $x_2$ 使用 $H_1$ 映射到位置 $H_1(x_2)$, $x_3$ 使用 $H_2$ 映射到位置 $H_2(x_3)$, $x_4$ 使用 $H_2$ 映射到位置 $H_2(x_4)$.

冲突处理: 当有新的数据元素 (图 3 中的 $x_5$) 要插入时, 它首先使用哈希函数 $H_2$ 映射到位置 $H_2(x_5)$. 然而, 这个位置已经被占用 (图 3 中显示有冲突). 为了解决冲突, 被占用位置上的数据元素 $x_2$ 会被 “踢出”, 并通过另一个哈希函数 $H_1$ 重新映射到新的位置 $H_1(x_2)$. 如果新的位置仍然有冲突, 这个过程会继续, 直到找到一个空的位置或者达到一定的迭代次数限制.

Pinkas 等人[14]通过实验分析出哈希函数个数 $k$ 和哈希表长度 $b$ 的最佳关系, 确定了无 *stash* 情况下哈希函数

个数和哈希表长度最小值 $b$ 的关系, $k=3$ 时 $b=1.27n$, $k=4$ 时 $b=1.09n$, $k=5$ 时 $b=1.05n$ ($n$ 为元素集合大小).

朴素哈希表 (simple hash) 使用 $k$ 个 $hash:\{0,1\}^*\to\{0,1\}^\lambda$ 函数 $H_1,\ldots,H_k$, 将数据元素 $x$ 映射到朴素哈希表中, 与布谷鸟哈希不同的是, 朴素哈希表每个 $bin$ 中可以存放多个映射元素, 以应对冲突问题, 如图 4 所示.

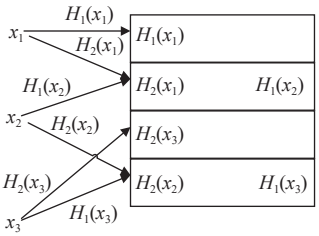



图 4 朴素哈希表示意图

在朴素哈希中, 有多个数据元素 (图 4 中的 $x_1$、$x_2$、$x_3$), 每个数据元素使用多个哈希函数 (图 4 中显示有两个哈希函数 $H_1$ 和 $H_2$) 映射到哈希表中的位置.

映射过程: 数据元素 $x_1$ 使用哈希函数 $H_1$ 映射到哈希表中 $H_1(x_1)$ 的位置, 同时也使用哈希函数 $H_2$ 映射到 $H_2(x_1)$ 位置. 数据元素 $x_2$ 使用哈希函数 $H_1$ 映射到位置 $H_1(x_2)$, 同时也使用哈希函数 $H_2$ 映射到位置 $H_2(x_2)$. $x_3$ 数据元素使用哈希函数 $H_1$ 映射到位置 $H_1(x_3)$, 同时也使用哈希函数 $H_2$ 映射到位置 $H_2(x_3)$. 朴素哈希的特点是简单直接, 当遇到哈希冲突的问题, 即不同的数据元素使用哈希函数映射到了相同的位置, 此时朴素哈希并不会“踢出”数据元素, 而是采用如链地址法 (如图 4 中所示) 的方式解决冲突.

### 2.2 弹性秘密共享

$(t,n)$ Shamir 秘密共享[35]是由秘密分发者选择一个秘密 $s$, 将秘密 $s$ 进行特定运算, 得到 $n$ 个秘密份额, 将 $n$ 个秘密份额分别交给 $n$ 个参与方保存, 当不少于 $t$ 个参与方同时拿出自己所拥有的秘密份额 $s_i$, $i\in[1,n]$, 即可还原出原始秘密 $s$. 在其基本形式中, $(t,n)$ Shamir 秘密共享要求所有参与方是诚实的, 并在重构秘密时提供正确的秘密份额. 在现实加密场景中, 通常需要防止参与方的恶意行为. 随着研究的深入, 一个秘密共享的强化版本被设计出来, 称为弹性秘密共享 (RSS)[27,36], 即使部分参与方输入了错误的份额, 只要提供的正确秘密份额数量达到门限值 $t$ 依然可以恢复秘密. 形式上, 所有参与方将他们的份额集中在一起, 当不少于 $t$ 个参与方提供正确的秘密份额, 即可还原原始秘密, 如图 5 所示. 弹性秘密共享正确性验证在第 3.2 节的引理 1 进行详细说明.

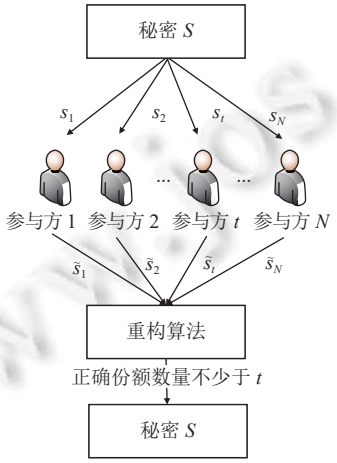



图 5 弹性秘密共享功能示意图

弹性秘密共享在实现过程中分为两个阶段: 一是秘密份额分享阶段, 二是秘密份额重构阶段. 本文基于拉格朗日插值和里德-所罗门解码分别设计了弹性秘密共享份额生成算法和弹性秘密份额重构算法, 如算法 1 和算法 2 所示. 与同态加密算法相比, 降低了计算复杂度.

**算法 1**. 弹性秘密共享份额生成算法 ($F_{\text{RSS-Gen}}$).

输入: 输入一个秘密值 $s$ 和需要生成的份额数量 $n$;
输出: $n$ 个秘密份额值 $s_i$, $i \in [1,n]$.

1. **for** $i = 1$ **to** $n$ **do** //生成随机值.
2. $rand()$ //选择一个随机值生成函数.
3. $r_i = rand()$
4. **end for**
5. 随机抽样 $t-1$ 个随机值 $\{r_1,\ldots,r_{t-1}\}$, 生成一个 $t-1$ 次多项式 $f(x) = s + r_1 x + \ldots + r_{t-1}x^{t-1}$, 使得 $f(0) = s$.
6. 随机选择 $n$ 个随机值 $\{x_1,\ldots,x_n\}$.
7. **for** $i = 1$ **to** $n$ **do** //计算秘密份额.
8. $c_i = f(x_i)$
9. **end for**
10. 输出 $s_i = c_i \| x_i$, $i \in [1,n]$.

**算法 2**. 弹性秘密份额重构算法 ($F_{\text{RSS-Rec}}$).

输入: 一个秘密份额集合 $s_i$, $i \in [1,n]$;
输出: 重构结果 $s'$.

1. 将 $n$ 个秘密份额 $s_i$, $i \in [1,n]$ 输入.
2. 使用里德-所罗门解码算法 $Decode(\cdot)$ (见引理 1), 解码得到一个多项式 $f'(\cdot)$.
3. $f'(\cdot) = Decode(s_i)$
4. 计算秘密 $s' = f'(0)$.
5. 输出 $s'$.

### 2.3 不经意键值对存储

不经意键值对存储 (oblivious key-value store, OKVS) 是一种数据结构[1], 用来隐藏数据中键值对的映射关系以及原始数据, 它由编码算法生成, 并可以通过解码算法进行查询.

编码算法 $F_{\text{Encode}}$: 输入一组 $\{(k_1,v_1),\ldots,(k_q,v_q)\} \in K \times V$, 其中 $K \times V$ 是有限键值字段, 并返回一个抽象数据结构 $D = F_{\text{Encode}}(K,V)$; 解码算法 $F_{\text{Decode}}$: 输入一个数据结构 $D$, 一个查询值 $k$, 返回一个值 $v = F_{\text{Decode}}(D,k)$. 当输入的查询值 $k = k_q$ 时, 则返回的值 $v = v_q$.

KVS 的正确性需要保证对于所有的 $M \in K \times V$, 其中密钥是不同的.

(1) $\Pr[F_{\text{Encode}}(M) = \perp]$ 可以忽略不计.

(2) 当 $F_{\text{Encode}}(M) = D \neq \perp$, 即编码成功. 使得存在 $(k,v) \in M$, 使得 $F_{\text{Decode}}(D,k) = v$, 即解码结果唯一.

**定义 1**. 对于一个概率多项式时间算法 $S$, 存在一个可忽略函数 $\mu(\lambda)$. 如果对于所有大小为 $m$ 的 $K_1$, $K_2$ 作为数据结构 $D$ 的输入, 输出的结果是不可区分的, 那么 KVS 是一个不经意键值对存储.

$$|\Pr[S(D,K_1)] - \Pr[S(D,K_2)]| < \mu(\lambda) \tag{1}$$

由于 OKVS 编码结构在通信过程中开销较小, 本文将 OKVS 与弹性秘密共享结合, 设计了一种高效的多参与方门限测试算法, 如算法 3 所示. 该算法具有较小的通信和计算开销.

**算法 3**. 多参与方门限测试算法 ($F_{\text{MTT}}$).

输入: 布谷鸟哈希表 $TC[i]$, OKVS 编码之后的数据结构 $D_j$ 和原始秘密 $H(s_i)$, 其中 $i$ 为哈希表的长度;
输出: 达到门限值的交集集合 $I$.

1. **for** $i = 1$ **to** $b$ **do**
2. **for** $j = 2$ **to** $N$ **do**
3. $s_{i,j} = F_{\text{Decode}}(D_j, TC[i])$ //秘密重构者通过 $TC[i]$ 中的数据元素解码数据结构 $D_j$, 得到 $i$ 组秘密份额.
4. **end for**
5. **end for**
6. **for** $i = 1$ **to** $b$ **do**
7. $s'_i = F_{\text{RSS-Rec}}(s_{i,1}, \ldots, s_{i,N})$ //输入 $i$ 组秘密份额到弹性秘密份额重构算法, 重构出 $i$ 个秘密.
8. **end for**
9. **for** $i = 1$ **to** $b$ **do**
10. **if** $H(s'_i) == H(s_i)$ // 当 $s'_i == s_i$ 时, 此时由 $TC[i]$ 解码出来的秘密份额与原始秘密相同.
11. insert $TC[i]$ to $I$ //此时 $TC[i]$ 中的数据元素为交集, 将其插入交集集合 $I$.
12. **else continue**
13. **end for**
14. **return** $I$ //输出交集集合 $I$.

### 2.4 单向散列函数

单向散列函数有一个输入和一个输出, 其中输入称为消息 (message), 输出称为散列值 (hash value). 单向散列函数可以根据消息的内容计算出散列值, 而散列值就可以被用来隐藏原始消息的内容.

首先, 我们将明确给出单向散列函数的数学定义: 设 $H(\cdot)$ 为一个单向函数, 其输入为 $x$, 输出为 $y$.

输入类型及范围: 输入 $x$ 属于集合 $X$, 其中 $X$ 可以是某一特定数据类型的集合, 例如 $X$ 可以是长度为 $n$ 比特的二进制字符串集合, 即 $X = \{0,1\}^n$, 或者是某一特定有限域上的元素集合等, 具体取决于协议所处理的数据类型.

输出类型及范围: 输出 $y$ 属于集合 $Y$, 通常 $Y$ 也是某一特定数据类型的集合. 当 $H(\cdot)$ 用于数据哈希, 输出集合 $Y$ 是长度为 $\lambda$ 比特的二进制字符串集合, 即 $Y = \{0,1\}^{\lambda}$.

在本文 $H(\cdot)$ 用于数据哈希, 其作用是将任意长度的数据 $x \in X$ 映射为固定长度的哈希值 $y \in Y$.

### 2.5 安全模型

本文在无合谋的半诚实模型下证明了协议的安全性, 假设所有协议参与者都在概率多项式时间内运行. 各方诚实地执行协议, 腐败的参与方不表现出主动的恶意行为, 也不偏离协议. 本文采用理想-现实模型进行安全性证明.

**定义 2**. 令 $f : (\{0,1\}^*)^N \to (\{0,1\}^*)^N$ 是一个确定性函数, $f_i$ 为安全计算协议 $\pi$ 的底层执行功能函数. $P_i$ 分别拥有输入和输出数据集 $X_i$ 和 $Y_i$, $C$ 为腐败的参与方集合, $\lambda$ 为安全参数.

现实模型 $View_C^{\pi}(\lambda, X_1, \ldots, X_N)$: $N$ 个参与方 $P_i$ 分别输入 $X_i$ 到协议 $\pi$ 中, 获得输出 $Y_i$. 即:

$$Y_i \leftarrow \pi(X_i) \tag{2}$$

理想模型 $Ideal_C^f(\lambda, (X_1, \ldots, X_N), f_C(X_1, \ldots, X_N))$: 模拟器 ($Sim$) 模仿现实模型中敌手视图, 控制腐败方的输入和输出.

安全性: 如果在多项式时间 $S$ 内, 对于任意 $C \subseteq [N]$ 满足:

$$Ideal_C^f\{S, (X_1, \ldots, X_N), f_C(X_1, \ldots, X_N)\}_{X \in (\{0,1\}^*)^N} \overset{c}{\equiv} View_C^{\pi}\{X_1, \ldots, X_N\}_{X \in (\{0,1\}^*)^N} \tag{3}$$

则认为协议在多项式时间内是安全的.

## 3 门限多方隐私集合交集协议

本节提出了一种基于弹性秘密共享的门限多方隐私集合交集协议, 协议中符号的定义见表 1. 根据第 2.1 节给出的布谷鸟哈希定义, 本文采取无 *stash* 的布谷鸟哈希表生成方式. 预处理阶段秘密分发者需要根据协议将自己的数据元素哈希到朴素哈希表并为每个 *bin* 关联对应的秘密份额. 在线阶段需要秘密重构者根据自己的数据元素去查询获得份额值进行重构, 协议流程图如图 6 所示, 具体协议如下.

表 1 协议符号表

| 符号 | 含义 | 符号 | 含义 |
|---|---|---|---|
| $t$ | 门限值 | $m \in [1,b]$ | 哈希表的索引 |
| $N$ | 参与方数量 | $s_m$ | 第 $m$ 个 *bin* 生成的秘密 |
| $n$ | 私有数据集大小 | $s_m^j$ | 发给参与方 $j$ 的秘密份额 |
| $\lambda$ | 计算安全参数 | $P_i$ | 参与方 $i$ |
| $\sigma$ | 统计安全参数 | $X_i$ | 参与方 $i$ 的私有数据集 |
| $k$ | 构造哈希表的哈希函数数量 | $D_i$ | 参与方 $i$ 编码的OKVS结构 |
| $h_k$ | 生成哈希表的哈希函数 | $TS_i$ | 参与方 $i$ 的朴素哈希表 |
| *bin* | 哈希表放置数据的位置 | $TC_i$ | 参与方 $i$ 的布谷鸟哈希表 |
| $b$ | 哈希表的长度 (*bin* 的数量) | $q$ | 朴素哈希表每个 *bin* 中的位置索引 |
| $H(\cdot)$ | 单向哈希函数 | $\eta$ | 秘密共享需要添加的冗余值数量 |
| $\mathbb{F}$ | 整数域 | *stash* | 布谷鸟哈希暂存区 |
| $f(\cdot)$ | 秘密共享插值出的多项式 | $I$ | 交集集合 |

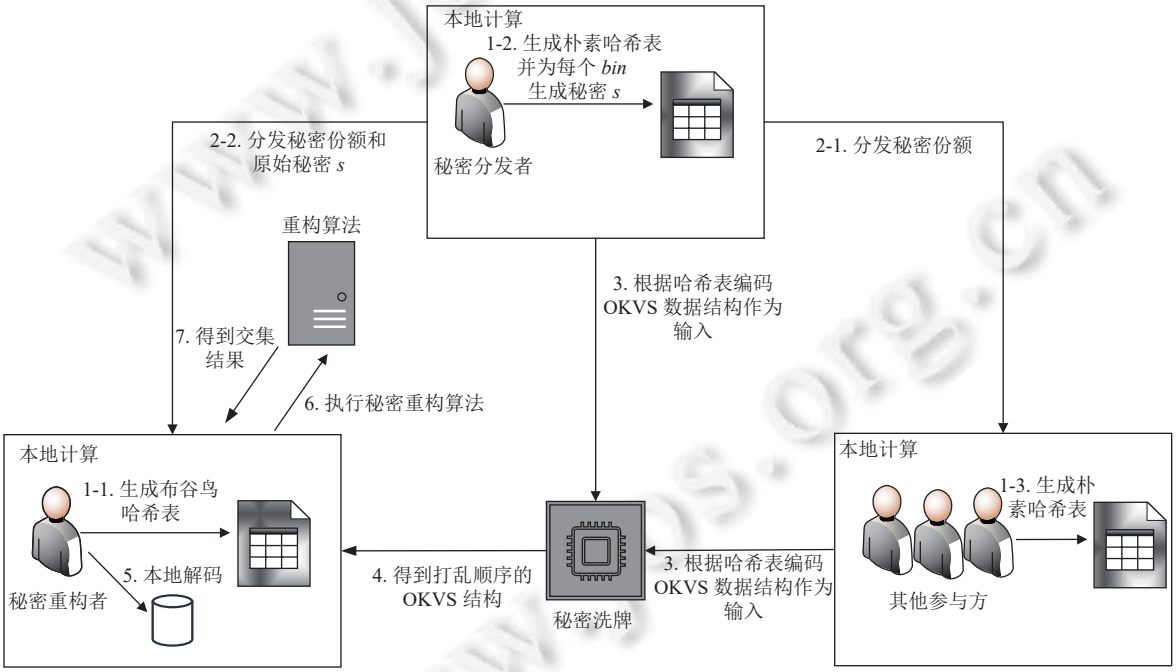



图 6 TMP-PSI 协议流程图

### 3.1 协议构造

**协议 1**. 基于弹性秘密共享的门限多方隐私集合交集协议 ($\Pi_{\text{TMP-PSI}}^{N,n,t}$).

参数: $N$ 个参与方 $P_i$, $i \in [1,N]$, 每个参与方的私有数据集合 $X_i$ 的大小为 $n$, 选择一个单向哈希函数 $H: \{0,1\}^* \to \{0,1\}^\lambda$, 门限值设置为 $t$, 所有参与方共同商议 $k$ 个哈希函数 $h_k: \{0,1\}^* \to \{0,1\}^\lambda$ 用于构造哈希表.

选择 $P_1$ 作为秘密重构参与方, $P_2$ 作为秘密分发参与方, $P_3,\ldots,P_N$ 作为普通参与方.

预处理阶段如下.

(1) $P_1$ 输入集合元素 $X_1$ 由 $k$ 个哈希函数生成布谷鸟哈希表 $TC_1$, 如图 6 步骤 1-1 所示.

(2) $P_i$, $i\in[2,N]$ 输入集合元素 $X_i$ 由 $k$ 个哈希函数生成朴素哈希表 $TS_i$. $P_2$ 作为秘密分发者为 $TS_2$ 中每个 $bin$ 生成秘密 $s$, 得到 $b$ 个秘密值 $s_m$, $m\in[1,b]$, 如图 6 步骤 1-2 和 1-3 所示. 为了保证每个秘密可以在任意门限值下正确重构, 要求在生成秘密份额时额外生成 $N-t$ 个冗余份额值 (在第 3.2 节的引理 1 中进行了详细说明), 因此 $P_2$ 需要将每个秘密 $s_m$ 和需要生成的份额数量 $2N-t$ ($N$ 个参与方和 $N-t$ 个冗余份额值) 执行 $F_{\text{RSS-Gen}}$ 秘密份额生成算法, 每个秘密对应插值出一个 $t-1$ 次多项式 $f_m(\cdot)$, 每个秘密对应 $2N-t$ 个秘密份额 $s_m\rightarrow\{s_m^1,\ldots,s_m^N,s_m^{N+1},\ldots,s_m^{2N-t}\}$.

在线阶段如下.

(1) $P_2$ 共享秘密份额 $s_m^j$, $m\in[1,b]$ 给参与方 $P_j$, $j\in[1,N]$, 如图 6 步骤 2-1 所示. 此外, 将冗余秘密份额 $\{s_m^{N+1},\ldots,s_m^{2N-t}\}$ 和原始秘密的哈希 $H(s_m)$ 发给 $P_1$, 如图 6 步骤 2-2 所示.

(2) $P_i$, $i\in[2,N]$ 将朴素哈希表 $TS_i[m]$ 中的每个元素 ($TS_i[m][q]$, $q$ 是朴素哈希表每个 $bin$ 中元素的位置索引) 与在线阶段步骤 (1) 中获得的秘密份额 $s_m^i$, $m\in[1,b]$ 异或, 即 $TS_i[m][q]\oplus s_m^i$, 构建一个新的键值对 $\{TS_i[m][q],TS_i[m][q]\oplus s_m^i\}$.

(3) $P_i$, $i\in[2,N]$ 输入自己的点集对执行 $F_{\text{Encode}}$, 编码一个 OKVS 结构 $D_i=F_{\text{Encode}}(TS_i[m][q],TS_i[m][q]\oplus s_m^i)$, 如图 6 步骤 3 所示. $P_i$, $i\in[2,N]$ 执行一个简单的洗牌算法, 打破数据结构 $D_i$ 与参与方之间的关系, 将洗牌之后的数据结构 $D_i$ 发给 $P_1$, 如图 6 步骤 4 所示.

(4) $P_1$ 本地解码数据结构 $D_i$, $i\in[2,N]$ 得到对应的秘密份额, 如图 6 步骤 5 所示.

(5) $P_1$ 将布谷鸟哈希表 $TC_1$ 和数据结构 $D_i$, $i\in[2,N]$ 输入多参与方门限测试算法 $F_{\text{MTT}}$, 输出交集 $I$, 如图 6 步骤 6 和 7 所示.

### 3.2 安全证明

**引理 1**. 弹性秘密共享的正确性验证.

证明: 在弹性秘密共享中, Reed-Solomon Codes 解码需要确保错误共享的数量不超过 $(n-t)/2$ [36], 因此需要添加 $\eta$ 个冗余值, 保证任意门限值下正确份额的数量满足 $t+\eta\geqslant t+(n+\eta-t)/2$, 因此 $\eta\geqslant n-t$, 在本文协议中参与方数量表示符号为 $N$, 因此协议中的冗余值为 $\eta\geqslant N-t$. 在下文的验证中依然采用 $n$ 进行普适性验证.

编码 Reed-Solomon Codes: 对于 $n$ 个抽样元素 $a_i\in\mathbb{F}_q$, $i\in[1,n]$, $q$ 是一个素数幂, 对包含 $k$ 个信息符号的数据包进行编码 $m_i\in\mathbb{F}_q$, 首先生成一个消息多项式 $f(x)$:

$$f(x)=m_1+m_2x+\ldots+m_kx^{k-1} \tag{4}$$

然后计算每个数据元素 $a_i$ 对应的多项式值 $c_i$:

$$c_i=f(a_i)\in\mathbb{F}_q,\ i\in[1,n] \tag{5}$$

得到对应的码字 $c$ 是:

$$c=(c_1,\ldots,c_n) \tag{6}$$

当信息符号的所有值在 $\mathbb{F}_q$ 上, 可以得到 $q^k$ 个 $n$ 长的码字. 很容易看到, 这些码字 $c$ 在 $\mathbb{F}_q$ 上形成一个线性码, 最小距离为 $d=n-k+1$ (这对任何 $(n,k)$ 线性码都是最优的). 这是由里德和所罗门提出的, 即所谓的 $(n,k,d)$ 里德-所罗门码.

解码 Reed-Solomon Codes: 获得一组来自码字 $c$ 的向量 $b=(b_1,\ldots,b_n)\in\mathbb{F}_q$, 其中包含了 $t$ ($t\leqslant(d-1)/2$) 个错误. 解码算法的目标是在多项式 $f(x)=m_1+m_2x+\ldots+m_kx^{k-1}$ 中找到定义了原始码字 $c$ 的消息多项式 $f(x)$. 需要预先计算多项式:

$$g_0=\prod_{i=1}^{n}(x-a_i)\in\mathbb{F}_q \tag{7}$$

在解码过程中可以找到一个多项式次数 $deg\leqslant n-1$ 的特殊多项式 $g_1(x)\in\mathbb{F}_q$, 使得:

$$g_1(a_i)=b_i,\ i\in[1,n] \tag{8}$$

然后应用扩展的欧几里得算法来实现 $g_0(x)$ 和 $g_1(x)$. 当剩余部分, 如 $g(x)$ 为 $deg\leqslant(n+k)/2$ 时停止. 假设现在有:

$$\begin{cases} g(x)=u(x)g_0(x)+v(x)g_1(x) \\ g(x)=v(x)f(x) \end{cases} \tag{9}$$

多项式 $v(x)$ 实际上是误差定位器多项式, 它的根包含了发生错误的所有位置 $a_i$.

$$u(x)g_o(x)=v(x)(f(x)-g_1(x)) \tag{10}$$

因此令 $v(x)(f(x)-g_1(x))=0$, 即 $f(x)-g_1(x)=0$. 用 $g(x)$ 除以 $v(x)$ 得到:

$$\frac{g(x)}{v(x)}=\frac{u(x)g_0(x)}{v(x)}+g_1(x) \tag{11}$$

变换形式得到:

$$\begin{cases} g(x)=f(x)v(x)+r(x) \\ degr(x)<degv(x) \end{cases} \tag{12}$$

如果 $r(x)=0$ 并且 $f(x)$ 满足 $deg<k$, 输出 $f(x)$. 否则解码失败, 此时意味着错误超过了 $(d-1)/2$.

快速傅里叶变换 (fast Fourier transform, FFT): Reed-Solomon Codes 解码需要高效的算法来实现多项式的乘法、插值等, 这些操作都可以通过快速傅里叶变换来实现.

例如多项式 $\left\{\begin{array}{l} f(x)=\sum_{i=0}^{n-1}a_ix^i \\ g(x)=\sum_{i=0}^{n-1}b_ix^i \end{array}\right\}$ 的乘积可以通过 $O(n^2)$ 的复杂度计算这个乘积的每一项的系数, 但 FFT 可以将多项式系数表示形式变换成点值表示形式, 以 $O(n\log n)$ 的时间复杂度来计算这个乘积.

对于 $n$ 次多项式可以代入 $n$ 个不同的点 $\{x_0,\dots,x_{n-1}\}$, 得到 $n$ 个值 $\{f(x_0),\dots,f(x_{n-1})\}$. 令 $f_p(x)$ 为多项式 $f(x)$ 的点值表示形式, 其点值形式表示为 $\{(x_0,f(x_0)),\dots,(x_{n-1},f(x_{n-1}))\}$.

$$fg_p(x)=\{(x_i,f(x_i)\cdot g(x_i)),\ i\in[0,n-1]\} \tag{13}$$

于是多项式的乘法在点值表示法下可以 $O(n^2)$ 的复杂度计算, 如果能够在较低的时间复杂度内将系数表示法转化为点值表示法, 再将点值表示法转回系数表示法, 就能以较低的时间复杂度计算多项式的乘法. 证毕.

**引理 2**. 对于任意整数 $s\geqslant1$, 存在一个足够大的常数 $a$, 插入所有元素项后存储的大小 $S$ 满足 $\Pr(S\geqslant s)=O(n^{-s})$.

证明: 对于分布 $D$, 设 $G(m,m,D)$ 表示双向图上的分布, 其中 $m=(1+\varepsilon)n$, $\varepsilon$ 是一个常数, 每个双向图上有 $m$ 个节点用 $v_i$, $i\in[1,m]$ 表示. 设 $T(G)$ 表示 $G$ 至少有一个循环的连接分量的数量, $f(G)$ 为 $G$ 中坏边的总数. $Po$ 表示满足泊松分布, 将 $B(u)$ 定义为连接 $u$ 到距离 $i-1$ 的节点的边数, 假设此时包含 $v$ 的连通分量的大小最多为 $k$. 意味着存在一个常数 $c\geqslant1$, 对于足够大的 $n$, 满足以下概率, 如公式 (14) 所示.

$$\begin{aligned} &\Pr(f(G(m,m,Po(\lambda)))-T(G(m,m,Po(\lambda)))\geqslant s) \\ &\leqslant\Pr\left(\sum_{i=1}^{2m}B_i'\geqslant s+|\{i:B_i'\geqslant1\}|\right)\leqslant\sum_{\substack{j_1,\dots,j_{2m}\\ \sum_{i=1}^{2m}j_i=s}}\prod_{\substack{i_1,\dots,i_{2m}\\ j_i\geqslant1}}\Pr(B\geqslant j_i+1)=\sum_{\substack{j_1,\dots,j_{2m}\\ \sum_{i=1}^{2m}j_i=s}}\prod_{\substack{i_1,\dots,i_{2m}\\ j_i\geqslant1}}cn^{-j_i-1} \\ &\leqslant\sum_{\substack{j_1,\dots,j_{2m}\\ \sum_{i=1}^{2m}j_i=s}}c^sn^{-s-|\{i:j_i\geqslant1\}|}\leqslant\sum_{k=1}^{2m}\binom{2m}{k}k^sc^sn^{-s-k}\leqslant\sum_{k=1}^{2m}\left(\frac{2(1+\varepsilon)n\mathrm{e}}{k}\right)^kk^sc^sn^{-s-k}=n^{-s}c^s\sum_{k=1}^{2m}\left(\frac{2(1+\varepsilon)\mathrm{e}}{k}\right)^kk^s=O(n^{-s}) \end{aligned} \tag{14}$$

布谷鸟哈希表的初始化, 本文引用了 Pinkas 等人[14]中的实验结果, 更加详细地证明请参考 Kirsch 等人[37]发表的论文. 哈希表的长度取决于哈希函数的数量 $k$ 以及暂存区 *stash* 的大小, 随着哈希函数的数量增多, 数据元素在哈希表的插入过程中可插入位置就增多, 所以哈希表的长度 $b$ 就减小, 另一方面, 暂存区 *stash* 的空间越大, 可以容纳的冲突元素就越多. 证毕.

**定理 1**. $\Pi_{\text{TMP-PSI}}^{N,n,t}$ 在半诚实场景下是安全的.

证明: 首先验证该协议在执行过程中的正确性, 然后证明该协议在半诚实模型下的安全性.

● 正确性. 份额分发阶段, 秘密重构者 $P_1$ 使用 $k$ 个哈希函数映射出布谷鸟哈希表 $TC_1$, 其他参与方 $P_i$, $i\in[2,N]$ 使用 $k$ 个哈希函数映射出朴素哈希表 $TS_i[m]$, $m\in[1,b]$. $P_1$ 需要在与其他参与方交互过程中得到 $t-1$ 个正确的份额即可重构秘密多项式 $f'(\cdot)$. 秘密重构者解密操作如下.

在线阶段, 当 $P_1$ 得到其余 $N-1$ 个参与方 $P_i$ 编码的数据结构 $D_i$, 将布谷鸟哈希表中的数据元素作为查询元素输入, 每个数据元素对应解码得到 $N-1$ 个份额值.

当 $TC_1[m]\subset TS_i[m]$, 即存在 $TS_i[m][q]=TC_1[m]$. 参与方 $P_1$ 可以通过查询得到正确的秘密份额值, 令 $rs_m^i$ 为解码 $D_i$ 得到的结果:

$$\begin{aligned} rs_m^i &= F_{\text{Decode}}(D_i, TC_1[m])\\ &= F_{\text{Decode}}(F_{\text{Encode}}(k_m^i, v_m^i), TC_1[m])\\ &= F_{\text{Decode}}(F_{\text{Encode}}(TS_i[m][q], TS_i[m][q]\oplus s_m^i), TC_1[m])\\ &= TS_i[m][q]\oplus s_m^i \end{aligned} \tag{15}$$

参与方 $P_1$ 异或自己的查询元素即可解出正确份额:

$$s_m^i = rs_m^i \oplus TC_1[m] \tag{16}$$

● 安全性. 证明协议 $\Pi_{\text{TMP-PSI}}^{N,n,t}$ 在半诚实模型下是安全的. 模拟器分别模拟腐败的秘密重构者 $P_1$, 腐败的秘密分发者 $P_2$ 和腐败的普通参与方 $P_i$, $i\in[3,N]$ 的视图.

第 1 种情况, 模拟器 ($Sim_1$) 模拟腐败的秘密重构者 $P_1$ 的视图. $Sim_1$ 收到来自 $P_1$ 的输入 $X_1$ 和安全参数 $\lambda$ 以及交集 $I$.

(1) 在预处理阶段, $Sim_1$ 根据 $P_1$ 的输入 $X_1$ 构造布谷鸟哈希表 $TC_1$.

(2) 在线阶段, $Sim_1$ 均匀随机采样 $b$ 个秘密 $s'_m$, $m\in[1,b]$ 以及 $2N-t$ 个随机点值执行弹性秘密共享生成算法, 计算秘密份额 $\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\}$ 和 $H(s'_m)$ 其中 $m\in[1,b]$.

(3) $Sim_1$ 根据交集 $I$ 均匀随机抽样 $n-|I|$ 个随机值 $X_r=\{x_1,x_2,\ldots,x_{n-|I|}\}$, 计算 $D'_i\leftarrow F_{\text{Encode}}(\{I,X_r\},\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\})$.

(4) $Sim_1$ 输出 $P_1$ 的视图为 $(X_1,\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\},H(s'_m),D'_i)$, 其中 $\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\}$ 和 $H(s'_m)$ 模拟的是真实协议中 $P_2$ 发送给 $P_1$ 的消息, $D'_i$, $i\in[2,N]$ 模拟的是真实协议中参与方 $P_i$ 发送给 $P_1$ 的消息.

模拟器模拟出的 $\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\}$ 与真实协议执行过程中 $P_2$ 随机抽样的 $\{s_m^1,\ldots,s_m^N,s_m^{N+1},\ldots,s_m^{2N-t}\}$ 是计算不可区分的. $H(s'_m)$ 模拟的是秘密分发者 $P_2$ 发送给重构者 $P_1$ 的原始秘密, 由于 $H(\cdot)$ 的单向不可逆性, 保证了重构方 $P_1$ 无法区分模拟器生成的秘密 $H(s'_m)$ 和真实协议执行过程中产生的 $H(s_m)$. $D_i$, $i\in[2,N]$ 是其他参与方 $P_i$ 发给重构方 $P_1$ 的 OKVS 数据结构, 由于 $P_1$ 没有其他参与方的输入, 基于 OKVS 编码的安全性保证了 $P_1$ 在理想世界获得的 $D'_i$, $i\in[2,N]$ 与真实世界获得的 $D_i$, $i\in[2,N]$ 是不可区分的. 因此秘密重构者理想世界的视图与现实世界视图是计算不可区分的, 即:

$$\{Sim_1(1^\lambda,X_1,I)\}\overset{c}{\equiv}\{View_1^\pi((X_1,\ldots,X_N),\lambda)\} \tag{17}$$

第 2 种情况, 模拟器 ($Sim_2$) 模拟腐败的秘密分发者 $P_2$ 的视图. $Sim_2$ 收到来自 $P_2$ 的输入 $X_2$ 和安全参数 $\lambda$.

(1) 在预处理阶段, $Sim_2$ 根据 $P_2$ 的输入 $X_2$ 构造朴素哈希表 $TS_2$.

(2) 在线阶段, $Sim_2$ 抽样一个洗牌规则 *seed* 执行洗牌算法, 获得输出 $D'_2$.

(3) $Sim_2$ 输出 $P_2$ 的视图为 $(X_2,D'_2)$.

$P_2$ 与其他诚实参与方进行洗牌算法, 只需证明输出值 $D'_2$ 的隐私性即可, 秘密洗牌算法中 *seed* 的选择是参与方私有的, $P_2$ 不知道其他参与方的洗牌规则使得 $P_2$ 无法区分模拟器在理想模型中随机生成的 $D'_2$ 和真实协议中执行打乱洗牌算法得到的 $D_2$. 因此秘密分发者 $P_2$ 的视图在理想世界与现实世界上计算不可区分. 即:

$$\{Sim_2(1^\lambda,X_2)\}\overset{c}{\equiv}\{View_2^\pi((X_1,\ldots,X_N),\lambda)\} \tag{18}$$

第 3 种情况, 模拟器 ($Sim_i$) 模拟腐败的普通参与者 $P_i$, $i\in[3,N]$ 的视图. $Sim_i$ 收到来自 $P_i$, $i\in[3,N]$ 的输入 $X_i$

和安全参数 $\lambda$.

(1) 在预处理阶段, $Sim_i$ 根据 $P_i$, $i\in[3,N]$ 的输入 $X_i$ 构造朴素哈希表 $TS_i$.

(2) 在线阶段, $Sim_i$ 均匀随机采样 $b$ 个秘密 $s'_m$, $m\in[1,b]$, 计算秘密份额 $\{s_m^{i'}\}$.

(3) $Sim_i$ 抽样一个洗牌规则 $seed$ 执行洗牌算法, 获得输出 $D'_i$.

(4) $Sim_i$ 输出 $P_i$, $i\in[3,N]$ 的视图为 $(X_i, s_m^{i'}, D'_i)$.

模拟器模拟出的 $\{s_m^{i'}\}$ 与真实协议执行过程中 $P_2$ 随机抽样的 $\{s_m^{i}\}$ 是计算不可区分的. $P_i$, $i\in[3,N]$ 与其他诚实参与方进行洗牌算法, 只需证明输出值 $D'_i$ 的隐私性即可, 秘密洗牌算法中 $seed$ 的选择是参与方私有的, $P_i$, $i\in[3,N]$ 不知道其他参与方的洗牌规则使得 $P_i$, $i\in[3,N]$ 无法区分模拟器在理想模型中随机生成的 $D'_i$ 和真实协议中执行打乱洗牌算法得到的 $D_i$. 因此秘密分发者 $P_i$, $i\in[3,N]$ 的视图在理想世界与现实世界上计算不可区分. 即:

$$\{Sim_i(1^\lambda, X_i)\}\overset{c}{\equiv}\{View_i^{\pi}((X_1,\ldots,X_N),\lambda)\} \tag{19}$$

## 4 拓展的门限多方隐私集合交集协议

### 4.1 协议构造

拓展的门限多方隐私集合交集协议是在 TMP-PSI 协议的基础上对秘密份额共享部分进行拓展, 实现了多参与方获得各自私有集合中达到交集的数据元素. 本节继续采用表 1 中定义的符号, 协议流程图如图 7 所示.

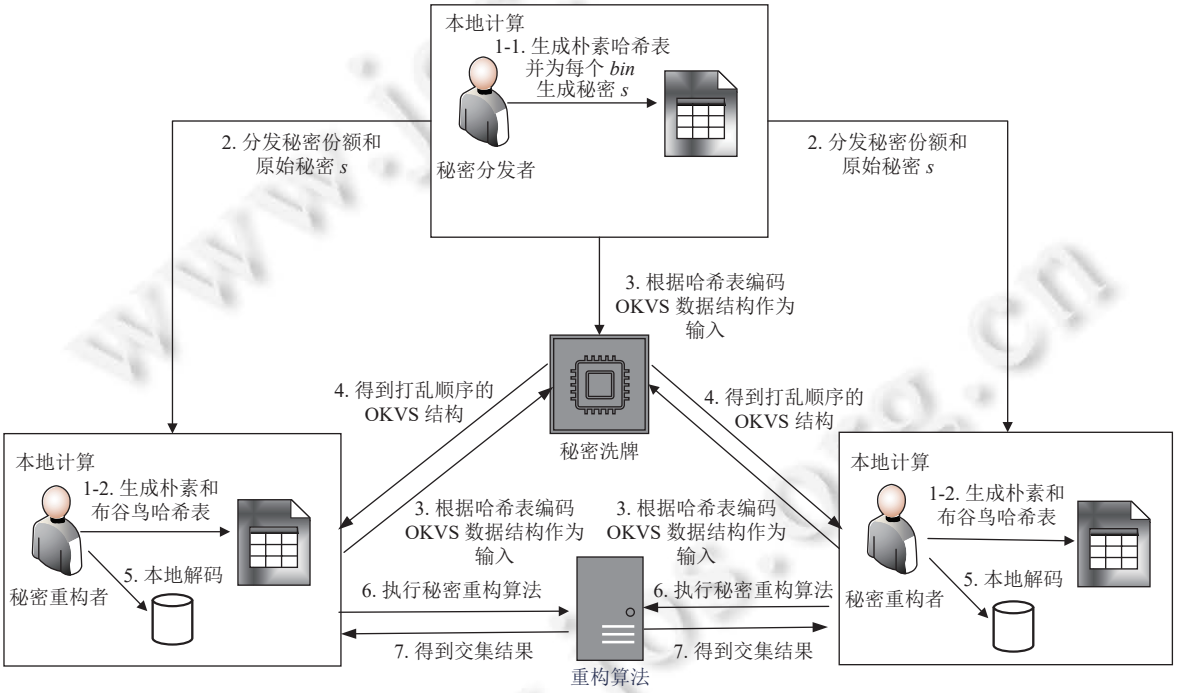



图 7 ETMP-PSI 协议流程图 (以 3 个参与方为例)

**协议 2**. 拓展的门限多方隐私集合交集协议 ($\Pi^{N,n,t}_{\text{ETMP-PSI}}$).

参数: 与协议 1 参数设置相同.

选择 $P_1$ 作为秘密分发参与方, $P_j$, $j\in[2,N]$ 作为秘密重构者, 拓展协议中没有普通参与方.

预处理阶段如下.

(1) $P_1$ 输入集合元素 $X_1$ 由 $k$ 个哈希函数生成朴素哈希表 $TS_1$. $P_1$ 作为秘密分发者为 $TS_1$ 中的每个 $bin$ 生成秘密 $s$, 得到 $b$ 个秘密值 $s_m$, $m\in[1,b]$ 并执行 $F_{\text{RSS-Gen}}$ 秘密份额生成算法, 每个秘密对应插值出一个 $t-1$ 次多项式

$f_m(\cdot)$, 每个秘密对应 $2N-t$ 个秘密份额 $s_m \to \{s_m^1,\ldots,s_m^N,s_m^{N+1},\ldots,s_m^{2N-t}\}$, 如图 7 步骤 1-1 所示.

(2) $P_j$, $j\in[2,N]$ 输入集合元素 $X_j$ 由 $k$ 个哈希函数生成朴素哈希表 $TS_j$ 和布谷鸟哈希表 $TC_j$, 如图 7 步骤 1-2 所示.

在线阶段如下.

(1) $P_1$ 将秘密份额 $\{s_m^j,s_m^{N+1},\ldots,s_m^{2N-t}\}$ 和 $H(s_m)$ 发给重构方 $P_j$, $j\in[2,N]$, 如图 7 步骤 2 所示.

(2) 所有参与方 $P_i$, $i\in[1,N]$ 将朴素哈希表 $TS_i[m]$ 中的每个元素与 $s_m^i$, $m\in[1,b]$ 异或, 即 $TS_i[m][q]\oplus s_m^i$, 其中 $TS_i[m][q]\leftarrow h_k(X_i)$ 并构建一个新的点集对 $\{TS_i[m][q],TS_i[m][q]\oplus s_m^i\}$.

(3) $P_i$, $i\in[1,N]$ 输入自己的点集对 $\{TS_i[m][q],TS_i[m][q]\oplus s_m^i\}$ 执行 $F_{\text{Encode}}$ 得到一个 OKVS 结构 $D_i=F_{\text{Encode}}(TS_i[m][q],TS_i[m][q]\oplus s_m^i)$, 如图 7 步骤 3 所示. $P_i$, $i\in[1,N]$ 执行一个简单的洗牌算法, 将洗牌之后的数据结构 $D_i$ 发给重构参与方 $P_j$, $j\in[2,N]$, 如图 7 步骤 4 所示.

(4) $P_j$, $j\in[2,N]$ 本地解码数据结构 $D_i$, $i\in[1,N]$ 得到对应的秘密份额, 如图 7 步骤 5 所示.

(5) $P_j$, $j\in[2,N]$ 将布谷鸟哈希表 $TC_j$ 和秘密份额输入多参与方门限测试算法 $F_{\text{MTT}}$ 得到各自的交集 $I_j$, 如图 7 步骤 6 和 7 所示.

### 4.2 安全证明

**定理 2**. $\Pi_{\text{ETMP-PSI}}^{N,n,t}$ 在半诚实场景下是安全的.

证明: 首先验证该协议在执行过程中的正确性, 然后证明该协议在半诚实模型下的安全性.

● 正确性. 预处理阶段, 每个秘密重构者 $P_i$, $i\in[2,N]$ 通过 $k$ 个哈希函数映射出布谷鸟哈希表 $TC_i$ 和朴素哈希表 $TS_i$. $P_i$ 可以通过 $TC_i[m]$ 表中的值, 在对应份额中找到 $N-t$ 个冗余值生成的对应秘密共享份额. $P_i$ 只需要在与其他参与方交互过程中得到 $t-1$ 个正确的份额即可重构秘密多项式 $f(\cdot)$. 秘密重构者解密操作如下.

在线阶段, 当 $P_i$ 获得其余 $N-1$ 个参与方编码的 OKVS 数据结构 $D_i$, 将布谷鸟哈希表中的数据元素作为查询元素输入, 每个数据元素对应得到 $N-1$ 个份额值. 正确性证明方式与定理 1 中的正确性证明方式相同.

● 安全性. 当秘密分发者 $P_1$ 不与其他秘密重构者 $P_i$, $i\in[2,N]$ 合谋的情况下, 此协议 $\Pi_{\text{ETMP-PSI}}^{N,n,t}$ 在半诚实模型下是安全的. 模拟器分别模拟腐败的秘密分发者 $P_1$, 腐败的秘密重构者 $P_i$, $i\in[2,N]$ 的视图 (依然采用第 3.1 节中的符号).

第 1 种情况, 模拟器 ($Sim_1$) 模拟腐败的秘密分发者 $P_1$ 的视图. $Sim_1$ 收到来自 $P_1$ 的输入 $X_1$ 和安全参数 $\lambda$.

(1) 在预处理阶段, $Sim_1$ 根据 $P_1$ 的输入 $X_1$ 构造朴素哈希表 $TS_1$.

(2) 在线阶段, $Sim_1$ 抽样一个洗牌规则 $seed$ 执行洗牌算法, 获得输出 $D_1'$.

(3) $Sim_1$ 输出 $P_1$ 的视图为 $(X_1,D_1')$.

$P_1$ 与其他诚实参与方进行洗牌算法, 只需证明输出值 $D_1'$ 的隐私性即可, 秘密洗牌算法中 $seed$ 的选择是参与方私有的, $P_1$ 不知道其他参与方的洗牌规则使得 $P_1$ 无法区分模拟器在理想模型中随机生成的 $D_1'$ 和真实协议中执行打乱洗牌算法得到的 $D_1$. 因此秘密分发者 $P_1$ 的视图在理想世界与现实世界上是计算不可区分的. 即:

$$\{Sim_1(1^\lambda,X_1)\}\overset{c}{\equiv}\{View_1^\pi((X_1,\ldots,X_N),\lambda)\} \tag{20}$$

第 2 种情况, 模拟器 ($Sim_i$) 模拟腐败的秘密重构者 $P_i$, $i\in[2,N]$ 的视图. $Sim_i$ 收到来自 $P_i$, $i\in[2,N]$ 的输入 $X_i$ 和安全参数 $\lambda$ 和交集 $I_i$.

(1) 在预处理阶段, $Sim_i$ 根据 $P_i$, $i\in[2,N]$ 的输入 $X_i$ 构造布谷鸟哈希表 $TC_i$ 和朴素哈希表 $TS_i$.

(2) 在线阶段, $Sim_i$ 均匀随机采样 $b$ 个秘密 $s_m'$, $m\in[1,b]$ 以及 $2N-t$ 个随机点值执行弹性秘密共享生成算法, 计算秘密份额 $\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\}$ 和 $H(s_m')$.

(3) $Sim_i$ 根据交集 $I_i$ 均匀随机抽样 $n-|I_i|$ 个随机值 $X_r=\{x_1,x_2,\ldots,x_{n-|I_i|}\}$, $Sim_i$ 可以模拟出其他参与方发送的消息 $D_j'\leftarrow F_{\text{Encode}}(\{I_i,X_r\},\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\})$, $j\in[1,N]$, $j\neq i$.

(4) $Sim_i$ 输出 $P_i$ 的视图为 $(X_i,\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\},H(s_m'),D_j')$, 其中 $\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\}$ 和 $H(s_m')$ 模拟的是秘密分发者 $P_1$ 发送给重构者的消息. $D_j'$ 模拟的是其他参与方 $P_j$, $i\in[1,n]$, $j\neq i$ 发送给 $P_i$ 的消息.

模拟器模拟出的 $\{s_m^{1'},\ldots,s_m^{N'},s_m^{N+1'},\ldots,s_m^{2N-t'}\}$ 与真实协议执行过程中 $P_1$ 随机抽样的 $\{s_m^1,\ldots,s_m^N,s_m^{N+1},\ldots,s_m^{2N-t}\}$ 是计算不可区分的. $H(\cdot)$ 的单向不可逆性, 保证了重构方无法区分模拟器生成的秘密和真实协议执行过程中产生的

$H(s_m)$. $D'_j$, $j\in[1,N]$, $j\neq i$ 是其他参与方 $P_j$ 发给 $P_i$ 的 OKVS 数据结构, 由于 $P_i$ 没有其他参与方的输入, 基于 OKVS 编码的安全性保证了 $P_i$ 在理想世界获得的 $D'_j$, $j\in[1,N]$, $j\neq i$ 与真实世界获得的 $D_j$, $j\in[1,N]$, $j\neq i$ 是不可区分的, 即 $\{D'_j\}=\{F(X_j,TS_j,s_m^j)\}$, $F$ 为真实协议执行中根据输入执行的算法. 因此秘密重构者理想世界的视图与现实世界视图是计算不可区分的. 即:

$$\{Sim_i(1^\lambda,X_i,I_i)\}\overset{c}{\equiv}\{View_i^\pi(\{X_1,\ldots,X_N\},\lambda)\} \tag{21}$$

## 5 实验结果与分析

本文设计的两种协议均使用 C++实现, 调用 GMP、NTL、LIBOTE 和 CRYPTO++等密码学库并在 Linux Ubuntu 20.04 系统上进行演示实验, 其中 CPU 为 AMD RYZEN7 6800H @ 3.20 GHz, 运行内存为 32 GB. 设置协议计算安全参数 $\lambda=128$, 统计安全参数 $\sigma=40$. 本文在实验部分选择构造哈希表的哈希函数数量与生成的哈希表长度为 $k=3$, $b=1.27n$. 为保证比对实验的准确性, 本文与其他文献的对比实验均在同一网络带宽环境下运行. 数据集大小、参与方数量等设置在第 5.3 节实验数据部分进行具体说明.

### 5.1 计算复杂度

第 1 种 TMP-PSI 协议中存在 3 类参与方 (秘密分发者, 秘密重构者和其他参与方). 预处理阶段, 秘密分发者生成朴素哈希表和秘密份额的计算复杂度为 $O(kn+bN)$, 在线阶段, 秘密分发者执行 OKVS 编码协议的计算复杂度为 $O(kn)$, 因此协议中秘密分发者的整体计算复杂度为 $O(2kn+bN)$. 预处理阶段, 秘密重构者生成布谷鸟哈希表其计算复杂度为 $O(n)$. 在线阶段, 秘密重构者执行 OKVS 解码算法的计算复杂度为 $O(nN)$, 执行 RSS 的重构算法复杂度为 $O(bN\log N)$, 因此协议中秘密重构者的整体计算复杂度为 $O(bN\log N+nN)$. 其他参与方生成朴素哈希表并执行 OKVS 编码协议, 其计算复杂度为 $O(2kN)$.

第 2 种 ETMP-PSI 协议, 协议中只有两类参与方 (秘密分发者和秘密重构者). 秘密分发者与秘密重构者的计算复杂度与 TMP-PSI 协议相同.

### 5.2 通信复杂度

在 TMP-PSI 协议在线阶段, 秘密分发者为其余 $N-1$ 个参与方发送秘密份额给其他参与方, 其通信复杂度与参与方数量 $N$ 和朴素哈希表的长度 $b$ 有关, 通信复杂度为 $O(b(N-1)\lambda)$. 重构时, 秘密分发者发送 OKVS 数据结构给重构方, 通信复杂度为 $O(b\lambda)$, 因此协议中秘密分发者的通信复杂度为 $O(bN\lambda)$. 在线阶段, 秘密重构者接收到秘密分发者发来的份额值, 其通信复杂度与哈希表的长度和冗余值的大小有关, 通信复杂度为 $O(b(N-t+1)\lambda)$. 重构时, 秘密重构者得到其余 $N-1$ 个参与方发送的信息, 其通信复杂度为 $O((N-1)b\lambda)$, 因此协议中秘密重构者的通信复杂度为 $O(b(2N-t)\lambda)$. 在线阶段, 其他参与方接收到秘密分发者发来的份额值, 通信复杂度为 $O(b\lambda)$. 重构时, 其他参与方发送 OKVS 数据结构, 通信复杂度为 $O(b\lambda)$, 因此协议中秘密分发者的通信复杂度为 $O(2b\lambda)$.

在 ETMP-PSI 协议的在线阶段, 秘密分发者为其余 $N-1$ 个参与方分发秘密份额, 秘密份额的数量与参与方数量 $N$ 以及朴素哈希表的长度 $b$ 有关, 此外每个 $bin$ 分发的份额数量为 $N-t+1$, 因此其通信复杂度为 $O(b(N-1)(N-t+1)\lambda)$. 重构时, 秘密分发者发送 OKVS 编码之后的数据结构, 通信复杂度为 $O(bN\lambda)$, 秘密分发者的总通信复杂度为 $O(bN^2\lambda)$. 在线阶段, 秘密重构者接收到秘密分发者发送的秘密份额, 通信复杂度为 $O(b(N-t+1)\lambda)$. 重构时, 秘密重构者得到其余 $N-1$ 个参与方的 OKVS 数据结构, 通信复杂度为 $O(b(N-1)\lambda)$, 因此协议中秘密重构者的通信复杂度为 $O(b(2N-t)\lambda)$.

### 5.3 实验数据

本节首先在参与方集合 $n=2^6$, $2^8$, $2^{10}$, $2^{12}$, $2^{14}$, $2^{16}$ 的设置下, 测试了两个协议在预处理阶段生成布谷鸟哈希表和朴素哈希表的运行时间, 如表 2 所示.

然后测试了数据集 $n=2^6$, $2^8$, $2^{10}$, $2^{12}$, $2^{14}$, $2^{16}$ 的设置下, 不经意键值对存储 (OKVS) 的运行时间和通信开销, 如表 3 所示.

表 2 预处理阶段哈希时间 (ms)

| 数据集大小 | 布谷鸟哈希 | 朴素哈希 |
|---|---|---|
| $2^6$ | 0.91 | 0.32 |
| $2^8$ | 1.46 | 0.74 |
| $2^{10}$ | 2.17 | 1.44 |
| $2^{12}$ | 3.55 | 2.42 |
| $2^{14}$ | 9.23 | 7.45 |
| $2^{16}$ | 27.03 | 15.81 |

表 3 OKVS 运行时间和通信开销

| 数据集大小 | 运行时间 (s) | 通信开销 (MB) |
|---|---|---|
| $2^6$ | 0.07 | 0.02 |
| $2^8$ | 0.11 | 0.09 |
| $2^{10}$ | 0.19 | 0.37 |
| $2^{12}$ | 0.32 | 1.49 |
| $2^{14}$ | 1.01 | 6.07 |
| $2^{16}$ | 2.33 | 24.30 |

本文基于弹性秘密共享设计了一种参与方门限测试方法. 在参与方数量 $N = 3, 4, 5, 7, 10$ 和参与方集合大小 $n = 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$ 的设置下, 测试不同门限下 RSS 算法的运行时间, 如图 8–图 10 所示, 分别为门限值等于 $0.3N$、门限值等于 $0.6N$ 和门限值等于 $0.8N$ 时 RSS 运行时间. 此外, 通过实验测试了随着参与方数量 $N$ 以及参与方集合大小 $n$ 的变化, OKVS 的运行效率的变化, 如图 11 所示.

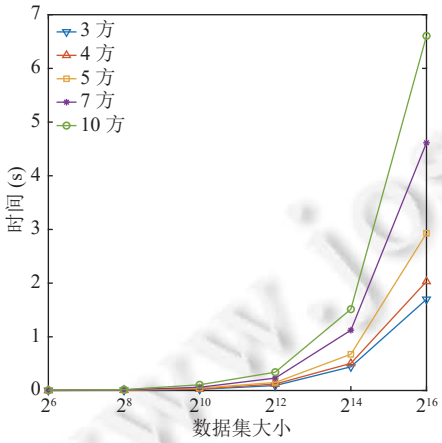



图 8 $t = 0.3N$ 时 RSS 运行时间

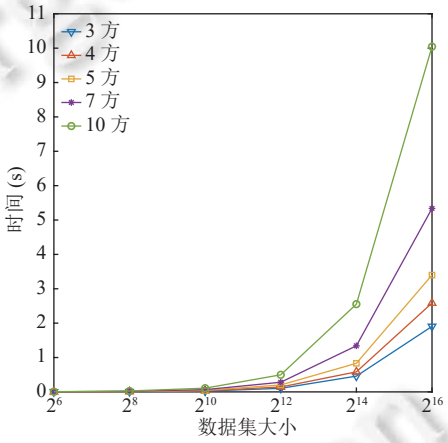



图 9 $t = 0.6N$ 时 RSS 运行时间

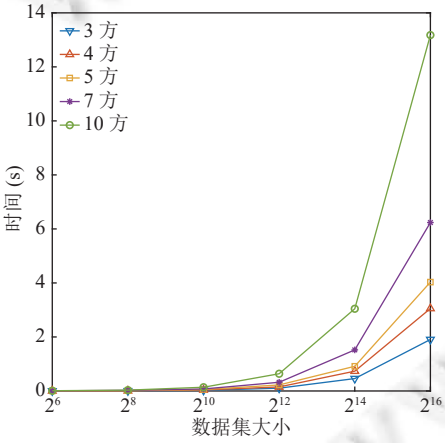



图 10 $t = 0.8N$ 时 RSS 运行时间

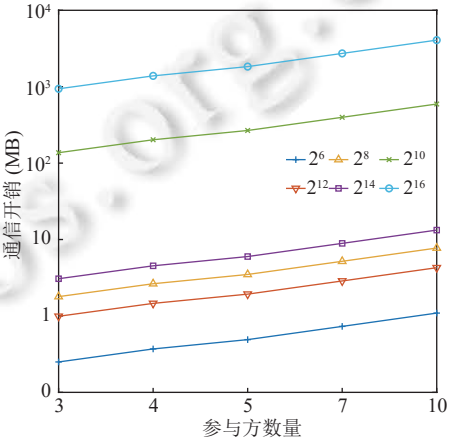



图 11 OKVS 通信开销 (多参与方)

基于上述组件, 测试了第 1 种 TMP-PSI 协议的整体运行时间以及通信开销, 本文设置实验梯度为参与方数量 $N = 3, 4, 5, 7, 10$ 和参与方集合大小 $n = 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$. 测试了在门限值 $t = 0.3N$, $t = 0.6N$, $t = 0.8N$ 的设置下协议的运行时间如表 4 所示.

表 4 TMP-PSI 协议整体运行时间 (s)

| 门限值 | 参与方数量 | $2^6$ | $2^8$ | $2^{10}$ | $2^{12}$ | $2^{14}$ | $2^{16}$ |
|---|---|---|---|---|---|---|---|
| $t=0.3N$ | 3 | 0.142 | 0.227 | 0.409 | 0.742 | 2.472 | 6.432 |
| | 4 | 0.213 | 0.340 | 0.605 | 1.093 | 3.044 | 9.111 |
| | 5 | 0.283 | 0.452 | 0.813 | 1.444 | 4.721 | 12.355 |
| | 7 | 0.425 | 0.678 | 1.211 | 2.171 | 7.184 | 18.731 |
| | 10 | 0.639 | 1.015 | 1.832 | 3.248 | 10.597 | 27.761 |
| $t=0.6N$ | 3 | 0.143 | 0.229 | 0.410 | 0.754 | 2.490 | 6.640 |
| | 4 | 0.214 | 0.341 | 0.611 | 1.115 | 3.122 | 9.665 |
| | 5 | 0.284 | 0.454 | 0.814 | 1.496 | 4.878 | 12.824 |
| | 7 | 0.426 | 0.684 | 1.216 | 2.221 | 7.402 | 19.454 |
| | 10 | 0.640 | 1.025 | 1.834 | 3.404 | 11.636 | 31.200 |
| $t=0.8N$ | 3 | 0.144 | 0.229 | 0.410 | 0.754 | 2.490 | 6.640 |
| | 4 | 0.215 | 0.341 | 0.614 | 1.123 | 3.272 | 10.133 |
| | 5 | 0.285 | 0.456 | 0.823 | 1.518 | 4.960 | 13.457 |
| | 7 | 0.428 | 0.690 | 1.225 | 2.262 | 7.582 | 20.354 |
| | 10 | 0.642 | 1.029 | 1.864 | 3.545 | 12.127 | 34.332 |

在 TMP-PSI 协议中通信开销与数据集大小和参与方数量相关性较大, 门限值的改变对协议的通信开销影响小, 因此仅测试了在门限值为 $t=0.3N$ 的设置下的总体通信开销如表 5 所示.

表 5 $t=0.3N$ 时 TMP-PSI 通信开销 (MB)

| 参与方数量 | $2^6$ | $2^8$ | $2^{10}$ | $2^{12}$ | $2^{14}$ | $2^{16}$ |
|---|---|---|---|---|---|---|
| 3 | 0.05 | 0.22 | 0.89 | 3.58 | 14.57 | 58.32 |
| 4 | 0.07 | 0.32 | 1.33 | 5.36 | 21.85 | 87.48 |
| 5 | 0.10 | 0.43 | 1.78 | 7.15 | 29.14 | 116.64 |
| 7 | 0.14 | 0.65 | 2.66 | 10.73 | 43.70 | 174.96 |
| 10 | 0.22 | 0.97 | 4.00 | 16.09 | 65.56 | 262.44 |

在相同实验梯度的设置下, 测试了第 2 种 ETMP-PSI 协议的运行时间和通信开销, 在门限值为 $t=0.3N$, $t=0.6N$, $t=0.8N$ 的设置下, 协议的运行时间如表 6 所示, 以及门限值 $t=0.3N$ 的设置下总体通信开销如表 7 所示.

表 6 ETMP-PSI 协议整体运行时间 (s)

| 门限值 | 参与方数量 | $2^6$ | $2^8$ | $2^{10}$ | $2^{12}$ | $2^{14}$ | $2^{16}$ |
|---|---|---|---|---|---|---|---|
| $t=0.3N$ | 3 | 0.213 | 0.338 | 0.601 | 1.066 | 3.481 | 8.786 |
| | 4 | 0.425 | 0.673 | 1.179 | 2.060 | 6.562 | 15.822 |
| | 5 | 0.707 | 1.117 | 1.959 | 3.375 | 10.748 | 25.191 |
| | 7 | 1.479 | 2.335 | 4.072 | 6.988 | 22.23 | 50.927 |
| | 10 | 3.167 | 4.986 | 8.688 | 14.79 | 46.67 | 107.03 |
| $t=0.6N$ | 3 | 0.214 | 0.340 | 0.602 | 1.080 | 3.499 | 8.994 |
| | 4 | 0.426 | 0.674 | 1.185 | 2.082 | 6.640 | 16.37 |
| | 5 | 0.708 | 1.119 | 1.960 | 3.427 | 10.90 | 25.66 |
| | 7 | 1.480 | 2.341 | 4.077 | 7.038 | 22.44 | 51.65 |
| | 10 | 3.168 | 4.996 | 8.690 | 14.95 | 47.70 | 110.46 |
| $t=0.8N$ | 3 | 0.214 | 0.340 | 0.602 | 1.080 | 3.499 | 8.994 |
| | 4 | 0.427 | 0.675 | 1.188 | 2.090 | 6.790 | 16.84 |
| | 5 | 0.709 | 1.121 | 1.969 | 3.449 | 10.98 | 26.29 |
| | 7 | 1.482 | 2.347 | 4.086 | 7.079 | 22.62 | 52.55 |
| | 10 | 3.170 | 5.009 | 8.720 | 15.093 | 48.20 | 113.60 |

本文的对比实验首先与文献 [17]、文献 [26] 和文献 [33] 比较协议的计算和通信复杂度, 如表 8 所示. 本文通过仿真实验对比协议整体性能, 如表 9 所示为 TMP-PSI 协议与文献 [17] 的对比实验结果, 由于文献 [17] 仅提供

了协议的整体运行时间, 没有提供通信开销, 因此本文与文献 [17] 仅在整体运行时间上进行了比较. 图 12 所示为 ETMP-PSI 协议与文献 [26] 和文献 [33] 的性能对比图, 图 12(a) 所示为在 $N = 10$, $n = 10^3$ 的设置下, 不同协议的计算开销比较, 图 12(b) 所示为在 $N = 10$, $n = 10^3$ 的设置下, 不同协议的通信开销比较.

表 7 $t = 0.3N$ 时 ETMP-PSI 通信开销 (MB)

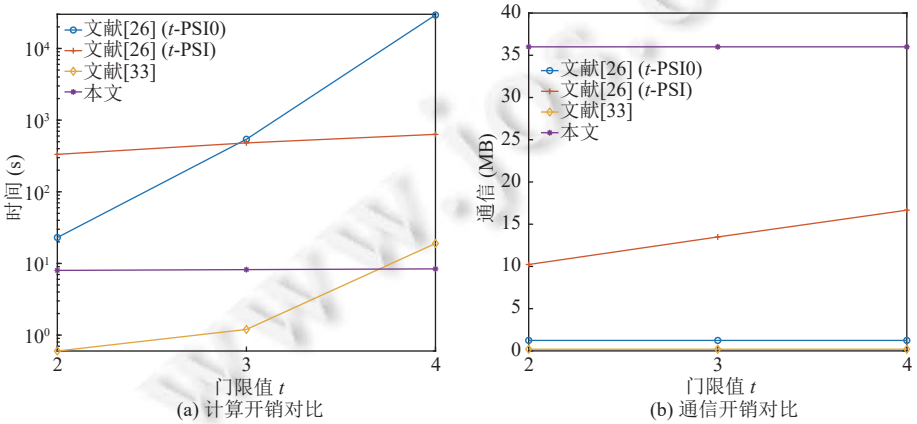
| 参与方数量 | $2^6$ | $2^8$ | $2^{10}$ | $2^{12}$ | $2^{14}$ | $2^{16}$ |
|---|---|---|---|---|---|---|
| 3 | 0.13 | 0.44 | 1.78 | 7.16 | 29.14 | 116.64 |
| 4 | 0.21 | 0.96 | 3.99 | 16.08 | 65.55 | 262.44 |
| 5 | 0.40 | 1.72 | 7.12 | 28.60 | 116.56 | 466.56 |
| 7 | 0.84 | 3.90 | 15.96 | 64.38 | 262.20 | 1 049.76 |
| 10 | 1.98 | 8.73 | 36.00 | 144.81 | 590.04 | 2 361.96 |

表 8 计算与通信复杂度比较

| 协议 | 通信轮数 | 秘密重构方 | | 秘密分发方 | | 其他参与方 | |
|---|---|---|---|---|---|---|---|
| | | 计算 | 通信 | 计算 | 通信 | 计算 | 通信 |
| 文献[17] | $t$ | $O(nN)$ | $O(nNt\log n\lambda)$ | $O(nN)$ | $O(nNt\log n\lambda)$ | — | — |
| 文献[26] ($t$-PSI0) | 2 | $O(n(N\log n/t)^{2t})$ | $O(nN\lambda)$ | $O(nN)$ | $O(nN\lambda)$ | $O(n)$ | $O(n\lambda)$ |
| 文献[26] ($t$-PSI) | 3 | $O(n(N\log n/t)^{2t})$ | $O(nN\lambda)$ | $O(nN)$ | $O(nNt\lambda)$ | $O(n)$ | $O(tn\lambda)$ |
| 文献[33] | 3 | $O(n(N\log n/t)^{2t})$ | $O(nN\lambda)$ | $O(nN)$ | $O(nN\lambda)$ | $O(n)$ | $O(n\lambda)$ |
| TMP-PSI | 2 | $O(bN\log N)$ | $O(b(2N-t)\lambda)$ | $O(bN)$ | $O(bN\lambda)$ | $O(kn)$ | $O(b\lambda)$ |
| ETMP-PSI | 2 | $O(bN\log N)$ | $O(b(2N-t)\lambda)$ | $O(bN)$ | $O(bN^2\lambda)$ | — | — |

表 9 TMP-PSI 协议性能比较 (s)

| 参与方数量, 门限值 | 协议 | $2^6$ | $2^8$ | $2^{10}$ | $2^{12}$ | $2^{14}$ | $2^{16}$ |
|---|---|---|---|---|---|---|---|
| $N = 3$, $t = 2$ | 文献[17] | 11.7 | 46.1 | 186.6 | 760.3 | — | — |
| | TMP-PSI | 0.14 | 0.23 | 0.41 | 0.75 | 2.49 | 6.64 |
| $N = 4$, $t = 3$ | 文献[17] | 15.2 | 72.6 | 297.1 | 1 123 | — | — |
| | TMP-PSI | 0.22 | 0.34 | 0.61 | 1.12 | 3.27 | 10.13 |
| $N = 5$, $t = 2$ | 文献[17] | 28.1 | 111.2 | 469.7 | — | — | — |
| | TMP-PSI | 0.28 | 0.45 | 0.81 | 1.44 | 4.72 | 12.36 |
| $N = 7$, $t = 5$ | 文献[17] | 50.7 | 214.7 | 832.5 | — | — | — |
| | TMP-PSI | 0.43 | 0.69 | 1.23 | 2.26 | 7.58 | 20.35 |
| $N = 10$, $t = 8$ | 文献[17] | 80.4 | 315.5 | 1 364.1 | — | — | — |
| | TMP-PSI | 0.64 | 1.03 | 1.86 | 3.55 | 12.13 | 34.33 |




(a) 计算开销对比 (b) 通信开销对比

图 12 ETMP-PSI 协议性能比较 ($N = 10$, $n = 10^3$)

## 6 总 结

本文针对现有 TMP-PSI 协议仅允许指定参与方获取交集信息以及计算开销大与通信轮数多的问题, 设计了一种兼顾通信轮数与计算开销的参与方门限测试方法, 提出了两种不同场景下的门限多方隐私集合交集方案. 第 1 种方案是指定参与方获取交集结果, 通过降低通信轮数和减少计算复杂度, 能有效提高运行效率. 第 2 种方案通过改变秘密份额分发方式, 使得秘密分发者在没有额外增加通信轮数和计算开销的同时, 实现了多参与方计算私有集合中交集信息的同时不泄露交集以外任何信息. 通过算法分析与实验结果表明, 所设计的协议与现有的方案相比拥有更好的性能.

## References


[1] Garimella G, Pinkas B, Rosulek M, Trieu N, Yanai A. Oblivious key-value stores and amplification for private set intersection. In: Proc. of the 41st Annual Int'l Cryptology Conf. Springer, 2021. 395–425. [doi: 10.1007/978-3-030-84245-1_14]

[2] Zhang E, Cai YQ. Rational secure two-party computation protocol. Journal of Computer Research and Development, 2013, 50(7): 1409–1417 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2013.20111614]

[3] Bui D, Couteau G. Improved private set intersection for sets with small entries. In: Proc. of the 26th IACR Int'l Conf. on Public-key Cryptography. Atlanta: Springer, 2023. 190–220. [doi: 10.1007/978-3-031-31371-4_7]

[4] Shi RH, Li YF. Quantum private set intersection cardinality protocol with application to privacy-preserving condition query. IEEE Trans. on Circuits and Systems I: Regular Papers, 2022, 69(6): 2399–2411. [doi: 10.1109/TCSI.2022.3152591]

[5] Rindal P, Schoppmann P. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In: Proc. of the 40th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Zagreb: Springer, 2021. 901–930. [doi: 10.1007/978-3-030-77886-6_31]

[6] He YY, Tan XY, Ni JB, Yang LT, Deng XJ. Differentially private set intersection for asymmetrical ID alignment. IEEE Trans. on Information Forensics and Security, 2022, 17: 3479–3494. [doi: 10.1109/TIFS.2022.3207911]

[7] Wang YH, Huang Q, Li HB, Xiao MY, Ma S, Susilo W. Private set intersection with authorization over outsourced encrypted datasets. IEEE Trans. on Information Forensics and Security, 2021, 16: 4050–4062. [doi: 10.1109/TIFS.2021.3101059]

[8] Qian YL, Shen J, Vijayakumar P, Sharma PK. Profile matching for IoMT: A verifiable private set intersection scheme. IEEE Journal of Biomedical and Health Informatics, 2021, 25(10): 3794–3803. [doi: 10.1109/JBHI.2021.3088289]

[9] Ling GW, Tang F, Cai CC, Shan JY, Xue HY, Li WL, Tang P, Huang XY, Qiu WD. $P^2$FRPSI: Privacy-preserving feature retrieved private set intersection. IEEE Trans. on Information Forensics and Security, 2023, 19: 2201–2216. [doi: 10.1109/TIFS.2023.3343973]

[10] Dörre F, Mechler J, Müller-Quade J. Practically efficient private set intersection from trusted hardware with side-channels. In: Proc. of the 29th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Guangzhou: Springer, 2023. 268–301. [doi: 10.1007/978-981-99-8730-6_9]

[11] Morales D, Agudo I, Lopez J. Private set intersection: A systematic literature review. Computer Science Review, 2023, 49: 100567. [doi: 10.1016/j.cosrev.2023.100567]

[12] Raghuraman S, Rindal P. Blazing fast PSI from improved OKVS and subfield VOLE. In: Proc. of the 2022 ACM SIGSAC Conf. on Computer and Communications Security. Los Angeles: ACM, 2022. 2505–2517. [doi: 10.1145/3548606.3560658]

[13] Pinkas B, Schneider T, Zohner M. Faster private set intersection based on OT extension. In: Proc. of the 23rd USENIX Conf. on Security Symp. San Diego: USENIX Association, 2014. 797–812.

[14] 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]

[15] 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]

[16] Ma M, Fu Y, Huang K, Jia XF. Light weighted privacy protection ViT inference framework based on secret sharing. Journal on Communications, 2024, 45(4): 27–38 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2024025]

[17] Bay A, Erkin Z, Hoepman JH, Samardjiska S, Vos J. Practical multi-party private set intersection protocols. IEEE Trans. on Information Forensics and Security, 2022, 17: 1–15. [doi: 10.1109/TIFS.2021.3118879]

[18] Kolesnikov V, Matania 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]

[19] Zhang E, Liu FH, Lai QQ, Jin GG, Li Y. Efficient multi-party private set intersection against malicious adversaries. In: Proc. of the 2019 ACM SIGSAC Conf. on Cloud Computing Security Workshop. London: ACM, 2019. 93–104. [doi: 10.1145/3338466.3358927]

[20] Wei LF, Liu JH, Zhang L, Wang Q, Zhang WJ, Qian XS. Efficient multi-party private set intersection protocols for large participants and small sets. Computer Standards & Interfaces, 2024, 87: 103764. [doi: 10.1016/j.csi.2023.103764]

[21] Ben-Efraim A, Nissenbaum O, Omri E, Paskin-Cherniavsky A. PSimple: Practical multiparty maliciously-secure private set intersection.

In: Proc. of the 2022 ACM on Asia Conf. on Computer and Communications Security. Nagasaki: ACM, 2022. 1098–1112. [doi: 10.1145/3488932.3523254]

[22] Yang YB, Dong XL, Cao ZF, Shen JC, Li RF, Yang YH, Dou SM. EMPSI: Efficient multiparty private set intersection (with cardinality). Frontiers of Computer Science, 2024, 18(1): 181804. [doi: 10.1007/s11704-022-2269-0]

[23] Nevo O, Trieu N, Yanai A. Simple, fast malicious multiparty private set intersection. In: Proc. of the 2021 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2021. 1151–1165. [doi: 10.1145/3460120.3484772]

[24] Chandran N, Dasgupta N, Gupta D, Obbattu SLB, Sekar S, Shah A. Efficient linear multiparty PSI and extensions to circuit/quorum PSI. In: Proc. of the 2021 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2021. 1182–1204. [doi: 10.1145/3460120.3484591]

[25] Kulshrestha A, Mayer J. Estimating incidental collection in foreign intelligence surveillance: Large-scale multiparty private set intersection with union and sum. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 1705–1722.

[26] Mahdavi RA, Humphries T, Kacsmar B, Krastnikov S, Lukas N, Premkumar JA, Shafieinejad M, Oya S, Kerschbaum F, Blass EO. Practical over-threshold multi-party private set intersection. In: Proc. of the 36th Annual Computer Security Applications Conf. Austin: ACM, 2020. 772–783. [doi: 10.1145/3427228.3427267]

[27] Zhang E, Qin LY, Yang RL, Li GL. Multi-party threshold private set intersection protocol based on robust secret sharing. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 5424–5441 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6743.htm [doi: 10.13328/j.cnki.jos.006743]

[28] Liu FH, Zhang E, Qin LY. Efficient multiparty probabilistic threshold private set intersection. In: Proc. of the 2023 ACM SIGSAC Conf. on Computer and Communications Security. Copenhagen: ACM, 2023. 2188–2201. [doi: 10.1145/3576915.3623158]

[29] Mohanty T, Srivastava V, Debnath SK, Das AK, Sikdar B. Quantum secure threshold private set intersection protocol for IoT-enabled privacy-preserving ride-sharing application. IEEE Internet of Things Journal, 2024, 11(1): 1761–1772. [doi: 10.1109/JIOT.2023.3291132]

[30] Ghosh S, Simkin M. The communication complexity of threshold private set intersection. In: Proc. of the 39th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2019. 3–29. [doi: 10.1007/978-3-030-26951-7_1]

[31] Badrinarayanan S, Miao PH, Raghuraman S, Rindal P. Multi-party threshold private set intersection with sublinear communication. In: Proc. of the 24th IACR Int'l Conf. on Public-key Cryptography. Springer, 2021. 349–379. [doi: 10.1007/978-3-030-75248-4_13]

[32] Ghosh S, Simkin M. Threshold private set intersection with better communication complexity. In: Proc. of the 26th IACR Int'l Conf. on Public-key Cryptography. Atlanta: Springer, 2023. 251–272. [doi: 10.1007/978-3-031-31371-4_9]

[33] Wei LF, Liu JH, Zhang L, Ning JT. Two cloud-assisted over-threshold multi-party private set intersection calculation protocol. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 5442–5456 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6747.htm [doi: 10.13328/j.cnki.jos.006747]

[34] Pagh R, Rodler FF. Cuckoo hashing. In: Proc. of the 9th Annual European Symp. on Algorithms. Aarhus: Springer, 2001. 121–133. [doi: 10.1007/3-540-44676-1_10]

[35] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612–613. [doi: 10.1145/359168.359176]

[36] Gao SH. A new algorithm for decoding Reed-Solomon codes. In: Communications, Information and Network Security. New York: Springer, 2003. 55–68. [doi: 10.1007/978-1-4757-3789-9_5]

[37] 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]

## 附中文参考文献

[2] 张恩, 蔡永泉. 理性的安全两方计算协议. 计算机研究与发展, 2013, 50(7): 1409–1417. [doi: 10.7544/issn1000-1239.2013.20111614]

[15] 李顺东, 周素芳, 郭奕旻, 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]

[16] 马敏, 付钰, 黄凯, 贾潇风. 基于秘密共享的轻量级隐私保护 ViT 推理框架. 通信学报, 2024, 45(4): 27–38. [doi: 10.11959/j.issn.1000-436x.2024025]

[27] 张恩, 秦磊勇, 杨刃林, 李功丽. 基于弹性秘密共享的多方门限隐私集合交集协议. 软件学报, 2023, 34(11): 5424–5441. http://www.jos.org.cn/1000-9825/6743.htm [doi: 10.13328/j.cnki.jos.006743]

[33] 魏立斐, 刘纪海, 张蕾, 宁建廷. 双云辅助的超阈值多方隐私集合交集计算协议. 软件学报, 2023, 34(11): 5442–5456. http://www.jos.org.cn/1000-9825/6747.htm [doi: 10.13328/j.cnki.jos.006747]

## 作者简介

**张恩**, 博士, 教授, CCF 高级会员, 主要研究领域为网络安全, 密码协议设计, 隐私保护.

**黄昱晨**, 硕士, 主要研究领域为密码协议, 安全多方计算.

**郑东**, 博士, 教授, 博士生导师, 主要研究领域为密码学理论与云安全.

**禹勇**, 博士, 教授, 博士生导师, 主要研究领域为公钥密码理论及应用, 区块链安全, 数据安全与隐私保护.