高效的隐私保护多方多数据排序

商 帅[1)] 李 雄[1),2),3)] 张文琪[1)] 汪小芬[1)]
李哲涛[4),5),6)] 张小松[1),3)]

[1)](电子科技大学计算机科学与工程学院(网络空间安全学院) 成都 611731)
[2)](四川大学数据安全防护与智能治理教育部重点实验室 成都 610065)
[3)](电子科技大学(深圳)高等研究院 广东 深圳 518110)
[4)](暨南大学网络安全检测与防护国家地方联合工程研究中心 广州 510632)
[5)](暨南大学数据安全与隐私保护广东省重点实验室 广州 510632)
[6)](暨南大学信息科学技术学院 广州 510632)

摘 要 安全多方计算允许具有私密输入的多个参与方联合计算一个多输入函数而不泄露各参与方私有输入的任何信息,因此近年来受到广泛关注.作为安全多方计算中的一个基础问题,隐私保护排序允许多个参与方在不泄露数据集隐私的前提下计算多个数据集的排序结果,广泛应用于产品定价、拍卖等场景.现有的隐私保护排序协议大多只支持两个参与方.而已有的多方多数据排序协议通信开销大、计算复杂度高,整体效率较低.现有隐私保护排序协议均未考虑恶意参与者的穷举攻击,因此安全保护不足.对此,本文提出一个高效的隐私保护多方多数据排序协议.多个参与方仅需$O(1)$轮交互即可以隐私保护的方式获得其持有的多个数据的排序结果.具体来讲,本文设计一种基于多项式的编码方法,将参与方的数据集编码为一个多项式,其每项的指数和系数分别代表数据和该数据的个数.通过多项式加法可实现多个参与方数据集的排序.同时,本文设计了多项式加密、聚合多项式生成和解密多项式生成算法,在保证计算正确性的同时实现多项式的隐私保护.最后,各参与方通过不经意传输技术获得排序结果.本文定义了不合谋参与方穷举攻击下的恶意安全.安全性分析表明本文协议不仅实现了半诚实安全性,而且达到了不合谋恶意用户穷举攻击的恶意安全性.此外,大量实验表明本文提出的协议在通信和计算方面都十分高效.如当参与方数量为15、每个参与方持有20 000个数据、数据上界为500 000时,本文协议的通信和计算开销分别为898.44 MB和69.76 s,仅为LDYW协议的12.08%和76.85%;而相对于AHM+方案,本文协议在通信开销仅增加约4倍的情况下使计算效率提升了约20倍.

关键词 隐私计算;安全多方排序;安全数据分析;隐私保护;排序
中图法分类号 TP391 **DOI号** 10.11897/SP.J.1016.2024.01832

Towards Efficient Privacy-Preserving Multi-Party Multi-Data Sorting

SHANG Shuai[1)] LI Xiong[1),2),3)] ZHANG Wen-Qi[1)] WANG Xiao-Fen[1)] LI Zhe-Tao[4),5),6)]
ZHANG Xiao-Song[1),3)]

[1)](*College of Computer Science and Engineering* (*College of Cyber Security*), *University of Electronic Science and Technology of China*, *Chengdu* 611731)
[2)](*Key Laboratory of Data Protection and Intelligent Management*, *Ministry of Education*, *Sichuan University*, *Chengdu* 610065)
[3)](*Shenzhen Institute for Advanced Study*, *University of Electronic Science and Technology of China*, *Shenzhen*, *Guangdong* 518110)
[4)](*National & Local Joint Engineering Research Center of Network Security Detection and Protection Technology*, *Jinan University*, *Guangzhou* 510632)
[5)](*Guangdong Provincial Key Laboratory of Data Security and Privacy Protection*, *Jinan University*, *Guangzhou* 510632)
[6)](*College of Information Science and Technology*, *Jinan University*, *Guangzhou* 510632)

收稿日期:2023-09-02;在线发布日期:2024-04-23.本课题得到国家自然基金项目(重点项目62332018,面上项目62072078和62271128)、四川省自然科学基金面上项目(2022NSFSC0550)、四川大学数据安全防护与智能治理教育部重点实验室开放课题(SCUSAKFKT202303Z)的资助.**商 帅**,博士研究生,主要研究领域为数据安全与隐私计算.E-mail:shshang180@gmail.com.**李 雄**(通信作者),博士,教授,主要研究领域为数据安全与隐私计算.E-mail:lixiong@uestc.edu.cn.**张文琪**,硕士研究生,主要研究领域为数据安全与隐私计算.**汪小芬**,博士,副教授,主要研究领域为应用密码学与安全协议.**李哲涛**,博士,教授,主要研究领域为计算机网络、人工智能及其安全.**张小松**,博士,教授,主要研究领域为区块链、人工智能安全等.

Abstract In recent years, secure multi-party computation has received extensive attention in academia and industry. Secure multi-party computation allows multiple participants with private inputs to jointly compute a multi-input function without revealing any information about the private inputs of each participant, which makes the data available but invisible. As a fundamental problem in secure multi-party computation, privacy-preserving sorting allows multiple participants to compute the joint sorting result of multiple datasets without disclosing the privacy of the datasets and the sorting result, which has a wide range of application needs and values, such as product pricing, auctions, interest recommendations. Most of the existing privacy-preserving sorting protocols only support two participants, and cannot meet the joint sorting requirements of multiple participants in practical scenarios. The existing multi-party multi-data sorting protocols have the problems of high communication overhead, high computational complexity. As a result, they all suffer low overall efficiency. At the same time, the existing privacy-preserving sorting protocols do not consider the malicious security model of exhaustive attacks by malicious participants, and only realize security under the semi-honest adversary model, thus providing insufficient security protection against the more realistic malicious adversary model. In this paper, we propose an efficient privacy-preserving multi-party multi-data sorting protocol to overcome the above problems. Through this protocol, multiple participants can collectively compute the sorting results of their data in a privacy-preserving way with only O(1) rounds of interaction. Specifically, this paper designs a polynomial-based encoding method that encodes a participant's dataset as a polynomial. In such a polynomial, the exponent and coefficient represent the data and the number of the data, respectively. Therefore, the sorting of multiple participant datasets can be realized by polynomial addition. For the above polynomial-based encoding algorithms, this paper also proposes polynomial encryption, aggregate polynomial generation, and decryption polynomial generation algorithms to realize the privacy protection of the encoded polynomials of each dataset based on the guarantee of computational correctness. The above algorithms can ensure the security of the encoded polynomials at the cost of low computational and communication overheads. Finally, each participant obtains the sorting result in a privacy-preserving way by means of communication-efficient oblivious transfer. In this paper, we consider a more realistic security model, provide for the first time the definition of malicious security under the non-colluding participant exhaustive attack, and define ideal functionalities of privacy-preserving sorting in different security models. The security analysis shows that the proposed protocol not only achieves semi-honest security, but also achieves malicious security against the exhaustive attack of non-colluding malicious users. In addition, a large number of experiments show that the proposed protocol is very efficient in both communication and computation. For example, when the number of participants is 15, each participant holds 20 000 data, and the upper bound of data is 500 000, our protocol's communication and computation overheads are 898.44 MB and 69.76 s, which are only 12.08% and 76.85% of that of the LDYW protocol. Compared with the AHM+ scheme, our protocol's computational efficiency is improved by about 20 times with only 4 times increase in the communication overhead.

Keywords privacy computing; secure multi-party sorting; secure data analysis; privacy protection; sorting

1 引 言

随着人们对隐私保护的重视程度越来越高，安全多方计算（Secure Multi-Party Computation，SMC）[1]成为学界研究的热点之一. SMC允许具有私密输入的多个参与方联合计算一个多输入函数，且在计算过程中不泄露各参与方私有输入的任何信息. 1982年，Yao[2]提出了第一个SMC协议以解决两方安全比较问题，即百万富翁问题. 自此，SMC的研究受到学者们的广泛关注，如隐私集合求交（PSI）[3-6]、隐私信息检索（PIR）[7-9]、不经意传输（Oblivious Transfer，OT）[10-12]以及隐私保护的多方排序[13-15]等. 其中，隐私保护的多方排序是一类重要的SMC协议，具有广泛的应用场景，如保密拍卖、匿名投票、安全数据挖掘、基于排名的推荐以及患者疾病严重程度的评估等. 如在患者疾病严重程度的评估场景下，假设三个医院分别收集了患者的医疗数据，包括患者的年龄和疾病严重程度（一个整数值，取值范围为[1,10]，值越高表示越严重）. 这三个医院希望在不透露患者具体的信息的情况下合作获得三院患者疾病严重程度的均值. 以上功能可通过隐私保护的多方排序来实现：

（1）每个医院首先对自己的数据进行局部排序，将患者按照疾病严重程度进行排序. 这个排序只在本地进行，不共享给其他医院.

（2）排序后，每个医院将排好序的数据使用安全多方计算协议在不泄露患者具体信息的情况下执行全局排序.

（3）多方计算协议对接收到的数据进行合并，以创建一个全局排序的列表，其中包含所有患者的疾病严重程度. 合并后的排序列表可以用于计算疾病严重程度的平均值，而不会暴露任何患者的具体信息. 每个医院只知道自己的数据在全局排序中的位置，但无法确定其他患者的信息.

具体来说，在隐私保护多方排序中，每个参与方以自己持有的待排序私有数据集为秘密输入，输出其私有数据集中每个数据在所有数据中的排序结果，而对其他参与方的私有数据集及对应的排序结果一无所知. 例如，在一个隐私保护的三方排序中，参与方P_1，P_2，P_3分别拥有私有数据集$D_1=\{203,514,639\}$，$D_2=\{36,723,871\}$和$D_3=\{402,639,871\}$. 每个参与方想要计算其私有数据集在D_1，D_2，D_3构成的有序超集$D=\{36,203,402,514,639,639,723,871,871\}$（假设为升序）中的排序. 运行隐私保护的三方排序协议后，P_1，P_2和P_3分别得到D_1，D_2和D_3在有序超集D中的排序结果$SR_1=\{2,4,5\}$，$SR_2=\{1,7,8\}$和$SR_3=\{3,5,8\}$.

随着大数据、云计算等技术的快速发展，现实应用场景对隐私保护排序技术在通信和计算性能以及安全性方面提出了更高的要求. 现有方案大都不具备高效的通信和计算性能，也未考虑更加实际的恶意敌手. 对此，本文采用实际的安全模型，提出轻量级数据编码技术和恶意用户行为检测方法，实现性能更高、安全性更强的隐私保护排序方案.

1.1 相关工作

实现隐私保护多方排序的最直接的方法是多次使用安全两方比较协议[2]来比较任意两个数据，从而确定每个数据的顺序. 然而，这种方法不仅会造成额外的信息泄露（如任意两个数据之间的大小关系），还会产生较大的通信和计算开销. 因此，为了实现多方隐私保护排序并提高其效率，国内外学者对此进行了深入的研究，我们对相关工作进行简要的回顾.

1983年，Ajtai等人[16]提出了一种被称为AKS排序网的渐进最优排序网络，它支持多数据的排序. 该排序网络在排序过程中主要通过“比较操作”来完成对多个数据的排序. 对于m个输入，该排序网络的“比较操作”复杂度为$O(m\cdot\log m)$. 但由于“比较操作”复杂度的常系数非常高，该算法并不适用于实际应用场景. Goldreich[17]提出一个支持多个数据的不经意数据排序协议，即随机希尔排序. 尽管随机希尔排序能高效地返回正确的排序结果，但该协议计算和通信开销都较大，还会泄露额外的隐私信息. 2007年，刘等人[18]在半诚实模型下设计了一个基于ElGamal同态密码的安全多方多数据排序协议. 2008年，肖等人[19]利用同态加密算法设计了半诚实安全的两方排序协议，并将该协议扩展到多方情形. 2009年，邱等人[20]基于RSA加密系统的同态性质设计了一个支持多方多数据的安全排序协议，但该协议仅实现半诚实模型下的安全，无法保证参与方恶意模型下的安全. 2014年，Bogdanov等人[21]提出了四种安全多方排序协议. 其中两种分别基于排序网络和快速排序，另外两种则是分别基于安全对比技术和不经意基数排序算法. 2017年，李等人[22]基于Paillier同态加密、椭圆曲线同态加密和秘密分割等技术提出了三个半诚实模型下安全的支持多个字符的安全多方排序协议. 为了避免大量的

比较操作，王等人[23]基于计数排序和桶排序的思想提出一种高效实用的安全多方排序. 该协议实现了安全多方排序的半诚实安全性和公平性等性质. Dai等人[24]提出了一个云辅助的安全多方排序协议. 该协议首先将明文加密得到对应的密文，之后借助逻辑图构造安全比较码，从而在保证隐私的同时实现对不同密文的比较. 2020年，李等人[25]设计了一个安全多方多数据排序协议. 在该协议中，每个参与方首先构建一个长度为n的密文向量，接收一个大小为m•n的密文矩阵，其中n和m分别为数据域上界和参与方数量. 在该协议中，每个参与方具有$O(m$•$n)$的巨大通信开销，不适用于数据域上界n较大时的排序场景. 因此，其实用性还有待加强.

另一方面，Chida和Hamada等人[26]提出了“理想通信模式”的概念，并且设计了一种能够支持理想通信模式的隐私保护的多方排序方案. 理想的通信模式是指该方案允许多个参与方之间无需直接通信即可完成排序. 为了实现这种理想通信模式，该方案首先对各个用户的数据集在3个服务器之间进行秘密共享. 之后，由3个服务器共同完成对所有数据的排序. 另外，Attrapadung等人[27]基于与文献[26]中的方案类似的方法构建了一个新的排序协议. 尽管这个排序协议在在线阶段只需要两个服务器就能实现与文献[26]中的方案相同的理想通信模式，但在离线阶段仍然需要三个服务器.

尽管现有工作在两方或多方隐私保护的排序方面取得一定的研究成果，但在功能、安全、效率等方面还存在如下问题：

(1)缺乏对多方、多数据的支持. 现有协议大多只支持两方多数据的排序，而在实际场景中，隐私保护的排序往往涉及多个参与方. 因此，如何实现支持多方、多数据的安全排序协议是一个待解决的问题.

(2)无法抵御恶意参与方的穷举攻击. 大多数现有协议都建立在半诚实安全模型下，而在实际场景中，恶意参与方往往可通过穷举攻击，即通过伪造一个包含数据域内所有数据的非法数据集来获取其他参与方的数据和排序结果. 因此，在保护参与者隐私的同时如何防范恶意参与方穷举攻击是当前的一个难题.

(3)通信和计算复杂度较高. 现有协议大多基于计算复杂度高的公钥密码设计，交互轮次多、通信和计算复杂度高、实用性较差. 因此，如何以较低的通信轮数复杂度实现通信和计算高效的安全排序是一个具有挑战性的问题.

1.2　主要思想

针对上述挑战，本文提出一个隐私保护的多方多数据排序协议. 该协议主要利用计数排序的思想来实现对多个数据集$D_1,D_2,\cdots,D_m$的升序或降序排序，即首先计算所有小于(大于)数据d_{ij}的数据的数量num_{ij}，从而得到d_{ij}的排序结果$num_{ij}+1$.

为了实现对多方多数据的支持，我们考虑将参与方包含$k(k\geqslant 1)$个私有数据的数据集$D_i=\{d_{i1},d_{i2},\cdots,d_{ik}\}$，其中$d_{i1},d_{i2},\cdots,d_{ik}\in[1,n]$，编码为多项式$P_i=\sum_{j=1}^{n}a_jx^j$，其中，数据$j\in[1,n]$被编码到$P_i$的第$j$次方项$x^j$，系数$a_j$表示数据集$D_i$中数据$j$的个数. 我们假设数据集$D_i$中不包含重复数据，因此，$a_j\in\{0,1\}$. 换句话说，对于任意$j\in[1,n]$，若$j\in D_i$，则$a_j=1$，否则，$a_j=0$. 据此，$m$个参与方的私有数据集可编码分别编码为多项式$P_1,P_2,\cdots,P_m$. 我们进一步对其进行聚合，得到聚合多项式$P_a=\sum_{i=1}^{m}P_i=\sum_{j=1}^{n}a_j'x_j$，其中，$a_j'=\sum_{l=1}^{m}a_l$. 显然，根据多项式的加法性质，在聚合多项式$P_a$中，数据$j$仍然被编码到第$j$次方项$x^j$，但该项系数表示的是$m$个数据集$D_1,D_2,\cdots,D_m$中数据$j$的总个数. 因此，根据计数排序的原理，对聚合多项式P_a中所有指数小于j的项的系数进行求和即可得到数据j的升序排序结果为$sr_j=1+\sum_{l=1}^{j-1}a_l'$.

为了实现隐私保护，我们考虑对编码多项式P_i进行加密. 具体来说，首先生成加密密钥向量BV_i，然后对P_i加密得到加密多项式EP_i. 通过发送加密多项式EP_i而非多项式P_i实现数据的隐私保护，但这同时也带来了新的问题：加密随机改变了多项式P_i的系数，从而改变了每个数据的个数在多项式P_i中的编码结果. 具体来说，假设数据j不在数据集D_i中，则多项式P_i中项x^j的系数编码为0. 然而在加密后该项系数可能被随机地加密为0或1. 也就是说，加密可能会改变数据的个数编码. 对于原来个数为0的数据j，加密后其个数编码可能为0或1. 基于计数排序，错误的数据个数编码将会导致参与方最终得到错误的排序结果. 为解决这一问题，我们根据每个参与方的编码多项式P_i的系数向量V_i和加密多项式EP_i的系数向量SV_i计算解密多项式系数向量DV_i，从而得到解密多项式DP_i. 利用DP_i消除加密多项式EP_i中数据个数编码错误导致的排序结果的错误；根据多项式的加法性质，聚合解密多项式$DP=\sum_{i=1}^{m}DP_i$可以消除聚合加密多项式$EP_a=\sum_{i=1}^{m}EP_i=\sum_{j=1}^{n}SV[j]$带来的排序结果错误. 最终，所有参与方可以隐私保护地获得各自数据集的排序结果.

为了实现对不合谋参与方的恶意穷举攻击行为进行检测，我们采用n选k不经意传输协议(Oblivious Transfer，OT)，每个用户根据自己的数据集构造一个选择集合Δ，并生成选择集合的证明信息，以便服务器对其选择集合Δ的大小进行验证，防止其恶意地构造大于其持有数据个数k的选择集合进而获得过多的排序结果. 若发现$|\Delta|>k$，则服务器立即返回中止消息，并终止协议执行，从而实现针对不合谋恶意参与方穷举攻击的安全性. OT协议虽然可防止恶意参与方的穷举攻击，但同时也带来了巨大的通信和计算开销. 换句话说，本文协议抵御恶意参与方穷举攻击是以牺牲巨大的通信和计算效率为代价的. 然而，尽管如此，本文协议仍然具有较好的通信效率和最优的计算效率.

1.3 主要贡献

基于上述思想，我们设计了一个高效的隐私保护多方多数据排序协议Π_{PMMS}. 本文主要贡献如下：

(1)设计了一种新的基于多项式的数据集编码方法. 通过该方法可将参与方的数据集以保序的方式编码为一个多项式，多项式每一项的指数和系数分别表示数据和该数据在数据集中的个数. 相应地，多个数据集可以通过对应多项式的加法来实现有序合并.

(2)根据设计的基于多项式的编码方法，提出一种隐私保护的多方多数据排序协议Π_{PMMS}. 每个参与方都能够以隐私保护的方式获得其私有数据的排序结果，但都无法获得其他参与方的数据及对应的排序结果.

(3)给出了标准模型下不合谋参与方穷举攻击下的恶意安全定义. 安全性分析表明Π_{PMMS}在半诚实模型下安全地实现了多方多数据排序，同时Π_{PMMS}也实现了不合谋恶意参与方穷举攻击下的恶意安全.

(4)方案实现了高效的通信和计算. 具体来说，当参与方数量为15、每个参与方持有20 000个数据、数据上界为500 000时，Π_{PMMS}的通信开销和计算开销分别为898.44 MB和69.76 s，仅为LDYW方案[25]的12.08%和76.85%；而相对于AHM+方案[27]，Π_{PMMS}在通信开销仅增加4倍(413.32%)的情况下计算效率提升了约20倍(4.86%).

1.4 内容安排

本文后续章节安排如下：第2节介绍方案所使用的基础知识. 第3节介绍协议Π_{PMMS}的系统模型、安全模型和设计目标. 第4节对本文设计的基于多项式的编码方法进行详细说明. 第5节详细介绍隐私保护的多方多数据排序协议Π_{PMMS}. 第6、7节分别对协议Π_{PMMS}进行正确性、安全性分析和性能评估. 最后，第8节总结全文.

2 基础知识和概念

本节简要介绍协议Π_{PMMS}所使用到的基础知识，即改进的n选k不经意传输协议[28]和Shamir加法秘密共享协议[29].

2.1 改进的n选k不经意传输(k-out-of-n OT)协议

不经意传输是构建隐私保护协议的重要密码学原语之一. 对发送方持有的n个秘密，k-out-of-n OT协议允许接收方以如下隐私保护的方式从中得到其选择的任意$k(k<n)$个秘密：

(1)接收方隐私：发送方*Sender*无法确定接收方*Receiver*选取了哪些消息.

(2)发送方隐私：接收方*Receiver*无法得到除了其选择的k个消息之外关于其他n-k个消息的任何信息.

Lai等人[28]提出了一种基于双线性群的通信高效的n选k不经意传输协议. 在原始协议[28]中，发送方需要发送一个包含n个群G_T元素的消息. 而在协议实例化时，G_T中的元素通常为1024-1932比特. 因此带来了较大的通信开销. 为了提升通信效率，我们采用一个哈希函数H：$G_T\rightarrow\{0,1\}^l$将群G_T中的元素映射为较短的、固定长度的字符串. 相比于原始协议[28]，改进协议可以节省75%-87%的通信开销. 我们将修改的OT协议命名为i-OT，其执行过程如下：

(1)参数设置：给定安全参数λ，服务提供商选择一个双线性映射e：$G\times G\rightarrow G_T$及群G的两个生成元g和h，其中群G和G_T的阶均为大素数p. 服务提供商随机选择一个随机数$\alpha\in Z_p^*$作为系统密钥，并计算$g_i=g^{\frac{1}{\alpha+i}}$，$h_i=h^{\alpha^i}$，$i\in[1,n]$. 此外，还额外选择一个哈希函数$H$：$G_T\rightarrow\{0,1\}^l$. 最后，系统参数$SP$为$\{e,g,h,g_1,g_2,\cdots,g_n,h_1,h_2,\cdots,h_n,H\}$.

(2)接收方消息生成：接收方持有1个选择集合$\Delta=\{l_1,l_2,\cdots,l_k\}$. 基于选择集合$\Delta$和系统参数$SP$，接收方随机选择$s\in Z_p^*$作为其私钥. 然后根据选择集合$\Delta$和$g_1,g_2,\cdots,g_n$由算法1[30]计算$P(\Delta)=g^{\frac{s}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}$并计算$\sum=h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)\alpha^{n-k}}{s}}$：

$$\Sigma$$
$$=(h_n\cdot h_{n-1}^{\sum_{j=1}^{k}l_j}\cdot h_{n-k+1}^{\sum_{j=1}^{k}\Pi_{t\in[1,k],t\neq j}l_t}h_{n-k}^{\prod_{j=1}^{k}l_j})^{\frac{1}{s}}$$
$$=h^{\frac{\alpha^{n-k}(\alpha^k+\sum_{j=1}^{k}l_j\alpha^{k-1}+\cdots+\sum_{j=1}^{k}\Pi_{t\in[1,k],t\neq j}l_t\alpha+\prod_{j=1}^{k}l_j)}{s}}$$
$$=h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)\alpha^{n-k}}{s}}$$

最后，接收方将消息集合$T=\{P(\Delta),\sum,k\}$发送给发送方.

(3)发送方消息生成：发送方持有一个秘密消息集合$M=\{m_1,m_2,\cdots,m_n\}$. 当收到接收方发来的消息集合T，发送方首先验证$e(P(\Delta),\sum)\overset{?}{=}e(g,h^{\alpha^{n-k}})$. 若验证失败，终止算法. 否则，发送方接受$|\Delta|\leqslant k$. 然后，随机选取$r\in Z_p^*$，并计算密文$C_0=P(\Delta)^r=g^{\frac{rs}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}$及$C_i=H(e(g^{\frac{1}{\alpha+i}},h)^r\|i)\cdot m_i,i\in[1,n]$. 最后，发送方将密文$C_i,i\in[0,n]$发送给接收方.

(4)接收方秘密恢复：当收到发送方发来的密文$C_i,i\in[0,n]$，接收方根据密文C_i、选择集合Δ、私钥s及系统参数SP计算$m_i=H(e(C_0,h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}{(\alpha+i)}})^{\frac{1}{s}}\|i)^{-1}C_i$. 最后，接收方获得秘密消息$m_i,i\in\{l_1,l_2,\cdots,l_k\}$.

算法1.　*Aggregate*.

输入：选择集合$\Delta=\{l_1,l_2,\cdots,l_k\}$和$g_i=g^{\frac{1}{\alpha+i}},i\in[1,n]$

输出：$P_{k-1,k}=g^{\frac{1}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}$

1. FOR $i=1$ TO k：
2. $P_{0,i}=g_i$
3. END
4. FOR $j=1$ TO k-1：
5. FOR $t=j+1$ TO k：
6. IF $l_j=l_t$
7. output $\perp$
8. ELSE
9. $P_{j,t}=(\frac{P_{j-1,j}}{P_{j-1,t}})^{\frac{1}{x_t-x_j}}$
10. END
11. END
12. OUTPUT $P_{k-1,k}$

2.2　Shamir秘密共享

Shamir (t,m)秘密共享协议[29]是一个门限秘密共享协议. 其允许m个参与方分别获得秘密S的一个秘密份额，只有当不少于t个参与方出示其秘密份额才能恢复秘密S.

本文考虑使用一个(m,m)-Shamir秘密共享协议. 该协议以一个秘密S为输入，为m个参与方分别分配一个加法秘密份额s_i，满足$S=\sum_{i=1}^{m}s_i$. 当所有参与方出示秘密份额才能恢复秘密S.

一个(m,m)-Shamir秘密共享协议包括两个部分：秘密分割和秘密重构.

(1)秘密分割：给定一个秘密S，选择一个常数项为S的m阶多项式f_m，为每个参与方U_i分配秘密份额$f_m(i)$. 之后，每个参与方计算自己持有的加法秘密份额$s_i=\prod_{1\leqslant j\neq i\leqslant n}\frac{-j}{(i-j)}f_m(i)$.

(2)秘密重构：根据m个参与方的秘密份额s_i，恢复秘密为$S=\sum_{i=1}^{m}s_i$.

3　模型和目标

本节描述协议Π_{PMMS}所使用的系统模型、安全模型，并给出协议Π_{PMMS}的设计目标.

3.1　系统模型

通常，云计算范式具有强大的通信、计算和存储能力，已成为外包计算的重要模式. 因此，本文考虑如图1所示的云辅助的多方排序模型，其主要包含两类实体，即云服务器S和m个参与排序任务的参与方$U_1,U_2,\cdots,U_m$. 所有参与方在云服务器的协助下以隐私保护的方式获得他们的数据在整个数据集中的排序结果.

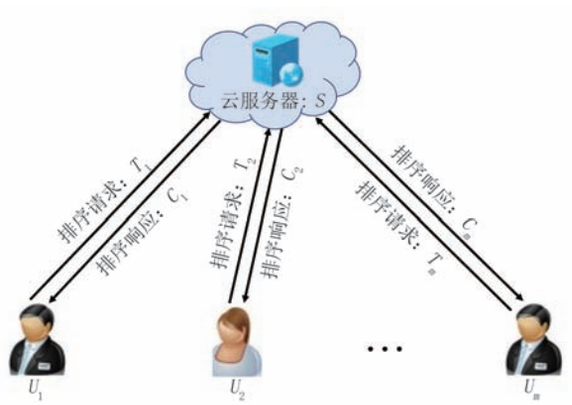


图1　系统模型

(1)参与方$U_1,U_2,\cdots,U_m$：每个参与方U_i拥有一个包含k个元素的数据集$D_i=\{d_{i1},d_{i2},\cdots,d_{ik}\}$，其中，$d_{ij}\in[1,n]$；对于$j\neq t$，有$d_{ij}\neq d_{it},i\in[1,m]$. 这里，我们假设所有参与方的隐私数据均为整数. 如果这些数据不是整数，则可通过正则化等技术将其归一化为整数. 为了计算数据集D_i中每个数据在所有参与方的数据集构成的有序超集D中的排序结果，U_i首先根据数据集D_i构造排序请求T_i，并将其发送至

云服务器. 当从云服务器接收到返回的加密排序结果后，U_i对其加密排序结果进行解密从而得到排序结果$SR_i=\{sr_{i1},sr_{i2},\cdots,sr_{ik}\}$.

(2)云服务器S：云服务器S承担一系列计算工作. 当收到参与方所有的排序请求$T_1,T_2,\cdots,T_m$后，S根据它们生成加密的排序结果$C_1,C_2,\cdots,C_m$. 最后S将加密的排序结果C_i分别返回给对应的参与方U_i. 参与方U_i通过将加密排序结果解密，恢复出其数据集D_i的排序结果SR_i.

3.2 安全模型

在协议Π_{PMMS}中，我们考虑两种安全模型：半诚实安全和恶意参与方穷举攻击下的恶意安全. 半诚实安全模型假设所有的实体都是半诚实的，即所有参与方都诚实地执行协议，但也会根据已有信息去推断其他参与方的隐私数据. 例如，用户U_i好奇其他参与方的隐私数据和对应的排序结果，云服务器想要推断出其他参与方的数据和排序结果. 而恶意参与方穷举攻击模型在半诚实安全模型基础上进一步考虑恶意敌手妥协参与方执行恶意穷举攻击的情况. 此外，在这两种安全模型下，我们假设参与方之间、参与方和服务器之间都没有合谋. 半诚实安全[31]和恶意参与方穷举攻击安全模型下的安全定义如下：

定义1. 半诚实安全. 一个协议Π针对静态半诚实敌手安全地计算了功能$\mathcal{F}$，若对于任何概率多项式时间(Probabilistic Polynomial-time, PPT)敌手A_i，存在PPT模拟器Sim_i，满足：

$(\text{Sim}_i(x_i,f_i(x_1,x_2,\cdots,x_m)))\overset{c}{\equiv}\text{view}_i^{\Pi}(x_1,x_2,\cdots,x_m))$，其中，$x_i$为第$i$个参与方的输入；$f_i(x_1,x_2,\cdots,x_m)$为第$i$个参与方的输出.

在上述半诚实安全定义基础上，我们进一步考虑参与者穷举攻击的恶意安全性[31-32]. 在协议Π_{PMMS}中，穷举攻击是指一个恶意参与方为了获取其他参与方的数据和排序结果，构造一个包含数据域中所有数据的全集$\{1,2,\cdots,n\}$作为自己的数据集. 以此，该参与方可以在协议结束时获得整个全集的排序结果，进而从排序结果恢复出其他参与方的数据及其排序结果. 比如，在一个三方隐私保护排序中，假设数据全集为$\{1,2,\cdots,8\}$，参与方P_1和P_2分别拥有数据集$D_1=\{1,5\}$和$D_2=\{3,7\}$. 为了恢复P_1和P_2的数据集，P_3构造任意大小的数据集$D_3=\{1,2,3,4,5,6,7,8\}$. 协议结束后，P_3得到排序结果为$\{1,3,4,6,7,9,10,12\}$，并由此可以判定P_1和P_2拥有数据1,3,5,7，而没有数据2,4,6,8. 这带来了极大的隐私泄露问题. 针对这种参与者穷举攻击，我们给出如下安全定义，称为恶意参与者穷举攻击下的恶意安全.

定义2. 不合谋参与方穷举攻击下的恶意安全. 在参与方不合谋的情况下，针对参与方U_i的恶意穷举攻击，协议Π中止安全地计算功能$\mathcal{F}$，若对于真实世界中任何敌手A_i，在理想世界中存在一个概率多项式时间(Probabilistic Polynomial-Time, PPT)的模拟器Sim_i，满足：

$(\text{IDEAL}^{\mathcal{F}}_i(x_i,f_i(x_1,x_2,\cdots,x_m)))\overset{c}{\equiv}\text{REAL}_i^{\Pi}(x_1,x_2,\cdots,x_m)$，其中，$\text{IDEAL}^{\mathcal{F}}_i(x_i,f_i(x_1,x_2,\cdots,x_m))$表示在理想世界中理想功能$\mathcal{F}_{\text{PMMS}}$执行的输出，$\text{REAL}_i^{\Pi}(x_1,x_2,\cdots,x_m)$表示在真实世界中协议$\Pi$执行的输出.

理想功能. 这里，我们给出隐私保护的安全多方多数据排序在理想安全定义下的形式化描述，即理想功能$\mathcal{F}_{\text{PMMS}}$：

参数:

(a)参与方U_i拥有的数据集D_i的大小为k；

(b)数据域上界为n.

功能:

(1)等待参与方U_i的输入$D_i=\{d_{i1},d_{i2},\cdots,d_{ik}\}$，如果$|D_i|>k$，则中止协议；

(2)返回D_i对应的排序结果$SR_i=\{sr_{i1},sr_{i2},\cdots,sr_{ik}\}$给参与方$U_i$.

3.3 设计目标

基于上述系统和安全模型，本文的目标是实现一个高效的隐私保护多方多数据排序协议Π_{PMMS}以解决现有方案存在的问题. 具体来讲，协议Π_{PMMS}应该达成以下目标：

(1)多方多数据排序：方案支持多方多数据排序，即对多个参与方、每个参与方持有多个数据，排序方案能为所有参与方输出其持有所有数据的正确排序结果；

(2)安全与隐私保护：方案不仅在半诚实模型下是安全的，并且能够实现参与方恶意穷举攻击下的恶意安全性，从而保护参与方的隐私数据集和对应排序结果的隐私，即云服务器和其他参与方都无法恢复出用户U_i的数据集D_i和排序结果SR_i；

(3)高效的通信和计算：与已有工作相比，本方案不仅需减少通信的交互轮次，同时需要大幅提高通信和计算效率.

4　基于多项式的编码方法

本节详细描述我们设计的基于多项式的编码方法以及相关算法. 这些算法是实现Π_{PMMS}协议的基础组件，包括四个算法：基础多项式编码算法*PolyEncode*、多项式加密算法*PolyEnc*、聚合加密多项式生成算法*PolyAgg*和解密多项式生成算法*PolyDec*.

4.1　基础多项式编码算法*PolyEncode*

当给定数据域$[1,n]$和取自该域的数据集$D_i=\{d_{i1},d_{i2},\cdots,d_{ik}\}$，基础多项式编码算法*PolyEncode*将$D_i$编码为一个$n$阶多项式$P_i$，详细步骤见算法2.

算法2.　*PolyEncode*.

输入：数据域$[1,n]$和数据集D_i

输出：多项式$P_i=\{n,V_i\}$

1. 根据数据集D_i生成一个长度为n的比特向量V_i，其中，对于$j\in[1,n]$，若$j\in D_i$，$V_i[j]=1$；否则，$V_i[j]=0$
2. 生成多项式$P_i=\sum_{j=1}^{n}V_i[j]\cdot x^j$，其中$V[j]$为多项式$P_i$的系数向量. 因此，可使用数据域上界$n$和长度为$n$的系数向量$V_i$构成的二元组$\{n,V_i\}$表示多项式$P_i$

图2给出了$n=10$，$D_i=\{2,3,5,8,10\}$时的多项式编码算法示例.

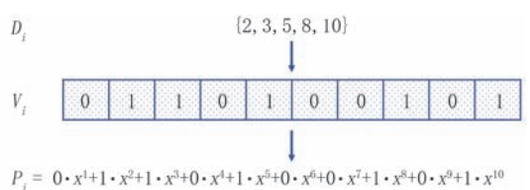


图2　多项式编码示例

4.2　多项式加密算法*PolyEnc*

对于由二元组$\{n,\boldsymbol{V}_i\}$表示的多项式P_i，其中n表示数据域上界，$\boldsymbol{V}_i$表示多项式P_i的系数向量，$\boldsymbol{V}_i[j]\in\{0,1\}$，$j\in[1,n]$，本算法首先生成一个长度为$n$的比特向量$BV_i$，将其作为密钥对多项式$P_i$进行加密得到加密多项式$EP_i$. 具体算法步骤如算法3所示.

算法3.　*PolyEnc*.

输入：多项式$P_i=\{n,\boldsymbol{V}_i\}$和密钥向量$BV_i$

输出：加密多项式$EP_i=\{n,\boldsymbol{EV}_i\}$

1. 生成n位比特向量BV_i，其每一位随机取自于$\{0,1\}$
2. 以BV_i为密钥，通过异或操作加密$\boldsymbol{V}_i$为$\boldsymbol{EV}_i$，即$\boldsymbol{EV}_i[j]=\boldsymbol{V}_i[j]\oplus BV_i[j]$
3. 输出多项式P_i对应的加密多项式$EP_i=\{n,\boldsymbol{EV}_i\}$

图3给出了*PolyEnc*算法的一个示例，其中输入为图2中的P_i.

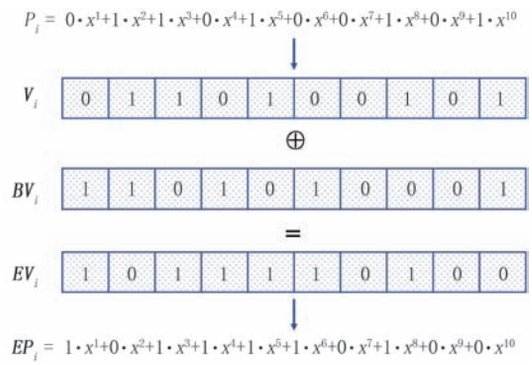


图3　多项式加密示例

需要注意的是，算法3的安全性证明与经典的One-time Pad加密一致，我们在此仅给出简要说明. 在算法3中，随机选择长度为n的比特向量BV_i，计算随机比特向量BV_i和多项式P_i长为n比特的系数向量的异或结果，将结果作为密文. 当接收到敌手给定的任意一对长为n比特明文m_0和m_1，预言机随机选取n比特的BV_i作为密钥k，并随机选择比特$b\in\{0,1\}$，计算$c=m_b\oplus k$并将c作为挑战密文返回给敌手. 显然，由于$k\in\{0,1\}^n$是一个随机字符串，$m_b\oplus k$在空间$\{0,1\}^n$中是均匀分布的. 所以，敌手无法区分密文c对应的明文是m_0还是m_1. 因此，多项式加密算法*PolyEnc*是安全的.

4.3　聚合加密多项式生成算法*PolyAgg*

本算法通过对加密多项式$EP_1,EP_2,\cdots,EP_m$执行聚合操作得到聚合加密排序多项式EP_a. 通过该算法得到的聚合加密排序多项式包含了最终的排序结果密文. 聚合加密多项式生成算法如算法4所示.

算法4.　*PolyAgg*.

输入：m个加密多项式$EP_1,EP_2,\cdots,EP_m$（$EP_i=\{n,EV_i\}$）

输出：聚合加密多项式EP_a

1. 基于m个元组$\boldsymbol{EV}_i$，生成整数向量$\boldsymbol{SV}$：

$$\boldsymbol{SV}[i]=\begin{cases}1, & i=1;\\ 1+\sum_{j=1}^{m}\sum_{l=1}^{i-1}\boldsymbol{EV}_j[l], & 2\leqslant i\leqslant n.\end{cases}$$

2. 输出聚合加密多项式$EP_a=\{n,\boldsymbol{SV}\}$

4.4　解密多项式生成算法*PolyDec*

输入加密多项式$EP_i=\{n,\boldsymbol{EV}_i\}$，$i\in[1,m]$，解密多项式生成算法输出解密多项式$\{DP_i\}_{i\in[1,m]}$（$DP_i=\{n,\boldsymbol{DV}_i\}$）和聚合解密多项式$DP=\{n,\boldsymbol{DV}\}$. 算法细节如算法5所示.

算法5. *PolyDec*.

输入：多项式$\{P_i\}_{i\in[m]}$的系数向量$\{\boldsymbol{V}_i\}_{i\in[m]}$和加密多项式$\{EP_i\}_{i\in[m]}$的系数向量$\{\boldsymbol{EV}_i\}_{i\in[m]}$

输出：解密多项式$\{DP_i\}_{i\in[m]}$($DP_i=\{n,\boldsymbol{DV}_i\}$)和聚合解密多项式$DP=\{n,\boldsymbol{DV}\}$

//阶段1：输出解密多项式$\{DP_i\}_{i\in[m]}$

1. 根据$\boldsymbol{V}_i$和$\boldsymbol{EV}_i$，按照如下方式构建一个大小为$n\times n$矩阵$\boldsymbol{M}_i$：

$$\boldsymbol{M}_i[j][l]=\begin{cases}\boldsymbol{V}_i[j]-\boldsymbol{EV}_i[j], & j<l\leqslant n\\ 0, & l\leqslant j\end{cases},$$

这里，$\boldsymbol{M}_i[j][l]$代表矩阵$\boldsymbol{M}_i$第j行、第l列的取值.

2. 计算解密多项式DP_i的系数向量$\boldsymbol{DV}_i$，即

$$\boldsymbol{DV}_i[l]=\sum_{j=1}^{n}\boldsymbol{M}_i[j][l]$$

3. 输出解密多项式$DP_i=\{n,\boldsymbol{DV}_i\}$

//阶段2：输出聚合解密多项式$DP=\{n,\boldsymbol{DV}\}$

4. 计算聚合解密多项式DP系数向量$\boldsymbol{DV}$，其中

$$\boldsymbol{DV}[j]=\sum_{i=1}^{m}\boldsymbol{DV}_i[j]$$

5. 输出聚合解密多项式$DP=\{n,\boldsymbol{DV}\}$

图4给出了一个10阶多项式P_{10}的聚合解密多项式生成示例，其中$P^{10}=0\cdot R^1+1\cdot R^2+1\cdot R^3+0\cdot R^4+1\cdot R^5+0\cdot R^6+0\cdot R^7+1\cdot R^8+0\cdot R^9+1\cdot R^{10}$.

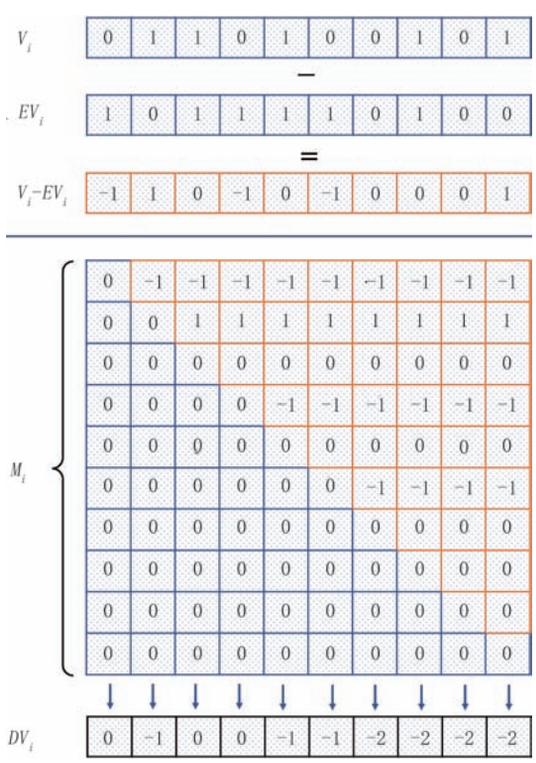


图4 解密多项式生成示例

5 排序方案Π_{PMMS}

本节介绍我们提出的隐私保护多方多数据排序方案Π_{PMMS}. 方案包含四个部分，即初始化、排序请求生成、加密多项式聚合和排序结果恢复. 表1列出了方案所使用的符号及其意义.

表1 符号及其意义

符号	描述
e	$G\times G\rightarrow G_T$的双线性映射
p	大素数，G和G_T的阶
g, h	群G的两个生成元
α	随机数$\alpha\in Z_p^*$
g_j, h_j	预计算值，$1\leqslant j\leqslant n$
n	数据域上界
m	参与方数量
U_i	第i个参与方
k	参与方的数据集大小
D_i	用户U_i的私有数据集
d_{ij}	数据集D_i中第j个数据，$1\leqslant j\leqslant k$
s_i	U_i的私钥
R	用户协商的整数
V_i	比特向量
BV_i	伪随机比特向量
EV_i	加密多项式系数向量，$EV_i=V_i\oplus BV_i$
P_i	由V_i生成的多项式
EP_i	由P_i生成的加密多项式
EP_a	由EP_i生成的聚合加密多项式
T_i	U_i的排序请求
SR_i	数据集D_i的排序结果密文
sr_{ij}^*	数据d_{ij}的排序结果密文
sr_{ij}	数据d_{ij}的排序结果

5.1 初始化

方案需执行下述步骤对系统进行初始化.

步骤1：输入安全参数1$^\lambda$，服务提供商选择一个双线性映射e：$G\times G\rightarrow G_T$、G的两个生成元g和h. 然后随机选择$\alpha\in Z_p^*$作为系统密钥，并计算$g_j=g^{\frac{1}{\alpha+j}}$，$h_j=h^{\alpha^j}$，$j\in[1,n]$. 此外，选择一个哈希函数$H$：$G_T\rightarrow\{0,1\}^l$.

步骤2：参与方$U_1,U_2,\cdots,U_m$协商n阶随机多项式$\Gamma=\sum_{j=1}^{n}\Gamma[j]x^j$，即$\Gamma[j]$代表多项式$\Gamma$的第$j$项的系数. 之后，参与方$U_1,U_2,\cdots,U_m$以多项式$\Gamma$为输入，联合执行2.2节所述的加法秘密共享协议. 协议结束后，参与方U_i，$i\in[1,m]$，获得多项式Γ的一个加法份额，即多项式γ_i，满足$\Gamma=\sum_{i=1}^{m}\gamma_i=\sum_{j=1}^{n}\Gamma[j]x^j$.

步骤3：公布公开系统参数$SP=\{e,g,h,g_1,g_2,\cdots,g_n,h_1,h_2,\cdots,h_n,H\}$.

5.2 排序请求生成

参与方U_i($1\leqslant i\leqslant m$)想要对其私有数据集D_i

执行隐私保护多方排序，按如下步骤生成排序请求 T_i（$1\leqslant i\leqslant m$）：

步骤1：U_i将数据集$D_i=\{d_{i1},d_{i2},\cdots,d_{ik}\}$作为选择集合$\Delta_i$，即$\Delta_i=D_i$. 之后，$U_i$选择一个随机数$s_i\in Z_p^*$，并基于选择集合$\Delta_i$和系统参数$SP$根据2.1节中的方法计算其选择集合的证明$P(\Delta)_i$和$\sum_i$：

$$P(\Delta)_i=g^{\frac{s_i}{(\alpha+d_{i1})(\alpha+d_{i2})\cdots(\alpha+d_{ik})}},$$

$$\sum_i=h^{\frac{(\alpha+d_{i1})(\alpha+d_{i2})\cdots(\alpha+d_{ik})\alpha^{n-k}}{s_i}}.$$

步骤2：以D_i和n为输入，参与方U_i通过调用基础多项式编码算法$PolyEncode$得到输出的编码多项式P_i.

步骤3：以P_i为输入，U_i继续调用多项式加密算法$PolyEnc$输出P_i的加密多项式EP_i.

步骤4：以P_i的系数向量V_i和EP_i的系数向量EV_i为输入，参与方U_i调用解密多项式生成算法$PolyDec$输出解密多项式DP_i. 然后，U_i计算混淆解密多项式$DP_i^*=DP_i+\gamma_i$.

步骤5：参与方U_i将排序请求$T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*\}$发送至云服务器$S$.

5.3　加密多项式聚合

当收到来自m个参与方的排序请求$T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*\}$，$1\leqslant i\leqslant m$，云服务器$S$执行下述步骤聚合加密多项式：

步骤1：基于输入T_i和系统参数SP，S通过检查$e(P(\Delta)_i,\sum_i)\stackrel{?}{=}e(g,h^{\alpha^{n-k}})$来执行OT协议的验证. 如果等式不成立，则中止协议，并发送Abort消息给各参与方；否则接收$|D_i|\leqslant k$，其中$1\leqslant i\leqslant m$.

步骤2：以$\{EP_i\}_{i\in[1,m]}$为输入，调用聚合加密多项式生成算法$PolyAgg$，得到聚合加密多项式EP_a. 同时，按照如下方式计算混淆的聚合解密多项式DP^*：

$$\begin{aligned}DP^*&=\sum_{i=1}^m DP_i^*\\&=\sum_{i=1}^m(DP_i+\gamma_i)\\&=\sum_{i=1}^m DP_i+\sum_{i=1}^m\gamma_i\\&=DP+\Gamma.\end{aligned}$$

步骤3：S将DP^*解析为$\{n,\boldsymbol{DV}^*\}$，其中，$\boldsymbol{DV}^*$是一个整数向量.

步骤4：S选择随机数$r\in Z_p^*$并计算密文$C_{i0}=P(\Delta)_i^r=g^{\frac{rs_i}{(\alpha+d_{i1})(\alpha+d_{i2})\cdots(\alpha+d_{ik})}}$和$C_{ij}=H(e(g^{\frac{1}{\alpha+j}},h)^r\|i)\cdot\boldsymbol{DV}^*[j]$，$1\leqslant j\leqslant n$.

步骤5：S将排序响应$C_i=\{C_{i0},C_{ij},EP_a\}$，$i\in[1,m]$，$j\in[1,n]$，发送给$U_i$.

5.4　排序结果恢复

当收到从云服务器S发来的排序响应$C_i=\{C_{0i},C_{ij},EP_a\}$，$U_i$执行如下步骤恢复数据集$D_i=\{d_{i1},d_{i2},\cdots,d_{ik}\}$的排序结果$SR_i=\{sr_{i1},sr_{i2},\cdots,sr_{ik}\}$，其中$sr_{ij}$代表数据$d_{ij}$的排序结果：

步骤1：U_i将EP_a解析为$EP_a=\{n,\boldsymbol{SV}\}$.

步骤2：U_i通过C_{i0}，C_{ij}，选择集合Δ_i，私钥s_i和系统参数SP，计算数据集D_i对应的混淆后的解密向量$\boldsymbol{DV}^*$：

$$\begin{aligned}\boldsymbol{DV}^*[d_{ij}]&=C_{d_{ij}}\cdot H(e(C_{i0},h^{\frac{\sum_{j=1}^k(\alpha+d_{ij})}{(\alpha+d_{ij})}})^{\frac{1}{s_i}}\|i)^{-1}\\&=C_{d_{ij}}\cdot H(e(P(\Delta)_i^r,h^{\frac{\sum_{j=1}^k(\alpha+d_{ij})}{(\alpha+d_{ij})}})^{\frac{1}{s_i}}\|i)^{-1}\\&=C_{d_{ij}}\cdot H(e(g^{\frac{r\cdot s_i}{\sum_{j=1}^k(\alpha+d_{ij})}},h^{\frac{\sum_{j=1}^k(\alpha+d_{ij})}{(\alpha+d_{ij})}})^{\frac{1}{s_i}}\|i)^{-1}\\&=C_{ij}\cdot H(e(g^{\frac{1}{\alpha+d_{ij}}},h)^r\|i)^{-1}\\&=H(e(g^{\frac{1}{\alpha+d_{ij}}},h)^r\|i)\cdot\boldsymbol{DV}^*[j]\cdot H(e(g^{\frac{1}{\alpha+d_{ij}}},h)^r\|i)^{-1},\end{aligned}$$

其中$d_{ij}\in D_i$，$1\leqslant j\leqslant k$.

步骤4：U_i解析$SR_i=\{sr_{i1},sr_{i2},\cdots,sr_{ik}\}$，即$sr_{ij}=sr[d_{ij}]=\boldsymbol{SV}[d_{ij}]+\boldsymbol{DV}^*[d_{ij}]-\Gamma[d_{ij}]$，其中$\Gamma[d_{ij}]$为多项式$\Gamma$的第$d_{ij}$项的系数，$1\leqslant i\leqslant m$，$1\leqslant j\leqslant k$.

为了便于理解，我们给出一个简略的隐私保护的多方排序实例，包括客户端排序请求生成实例、服务器端加密多项式聚合实例和客户端排序结果恢复实例.

客户端排序请求生成实例. 假设共有3个参与方U_1,U_2,U_3，即$m=3$，每个参与方拥有大小均为$k=3$的数据集D_1,D_2,D_3，数据域为$[1,7]$，即$n=7$. 其中，$D_1=\{1,4,6\}$，$D_2=\{2,3,6\}$，$D_3=\{1,3,5\}$. 按照初始化协议，参与方生成系统参数SP. 在排序请求生成阶段，U_i将数据集D_i作为选择集合Δ_i，即$\Delta_i=D_i$. 参与方U_1,U_2,U_3协商一个随机多项式$\Gamma=\sum_{j=1}^7\Gamma[j]x^j$，$\Gamma[j]$代表多项式$\Gamma$的第$j$项的系数. 之后，参与方$U_1,U_2,U_3$以多项式$\Gamma$为输入，联合执行2.2节中的Shamir加法秘密共享协议. 协议结束后，U_1,U_2,U_3获得多项式Γ的一个加法份额，即多项式γ_1,γ_2和γ_3，满足$\Gamma=\sum_{i=1}^3\gamma_i=\sum_{j=1}^7\Gamma[j]x^j$. 之后，$U_i$选择一个随机数$s_i\in Z_p^*$，并基于选择集合$\Delta_i$和系统参数$SP$根据2.1节中的方法计算其选择集合的证明$P(\Delta)_i$和$\sum_i$：

$$P(\Delta)_1=g^{\frac{s_1}{(\alpha+1)(\alpha+4)(\alpha+6)}},\sum_1=h^{\frac{(\alpha+1)(\alpha+4)(\alpha+6)\alpha^{n-3}}{s_1}},$$

$P(\Delta)_2=g^{\frac{s_2}{(\alpha+2)(\alpha+3)(\alpha+6)}}$, $\sum_2=h^{\frac{(\alpha+2)(\alpha+3)(\alpha+6)\alpha^{n-3}}{s_2}}$,

$P(\Delta)_3=g^{\frac{s_3}{(\alpha+1)(\alpha+3)(\alpha+5)}}$, $\sum_3=h^{\frac{(\alpha+1)(\alpha+3)(\alpha+5)\alpha^{n-3}}{s_3}}$.

U_1,U_2,U_3分别通过*PolyEncode*算法生成编码多项式$P_1=x^1+x^4+x^6$, $P_2=x^2+x^3+x^6$, $P_3=x^1+x^3+x^5$. 之后,U_1,U_2,U_3分别以P_1,P_2,P_3为输入调用*PolyEnc*算法(假设生成的密钥向量分别为$\boldsymbol{BV}_1=0101101$, $\boldsymbol{BV}_2=1001011$, $\boldsymbol{BV}_3=0010101$). *PolyEnc*分别输出对应的加密多项式为$EP_1=x^1+x^2+x^5+x^6+x^7$, $EP_2=x^1+x^2+x^3+x^4+x^7$, $EP_3=x^1+x^7$.

然后,根据P_1,P_2,P_3和EP_1,EP_2,EP_3,参与方U_1,U_2,U_3分别调用解密多项式生成算法*PolyDec*计算解密多项式$DP_1=(-x^3)+(-x^4)+(-x^6)+(-x^7)$, $DP_2=(-x^2)+(-x^3)+(-x^4)+(-2x^5)+(-2x^6)+(-x^7)$, $DP_3=(x^4)+(x^5)+(2x^6)+(2x^7)$. 之后,$U_i$计算混淆解密多项式$DP_i^*=DP_i+\gamma_i$. 最后,参与方$U_i$将排序请求$T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*)\}$发送给服务器$S$.

服务器端加密多项式聚合实例. 当收到3个参与方的排序请求$T_i=\{P(\Delta)_i,\sum_i,k=3,EP_i,DP_i^*\}$, $1\leqslant i\leqslant 3$,云服务器S执行下述步骤聚合加密多项式. S通过检查$e(P(\Delta)_i,\sum_i)\stackrel{?}{=}e(g,h^{\alpha^{n-3}})$来执行$i$-OT协议的验证. 如果任意一个等式不成立,则中止协议,并发送Abort消息给各参与方;否则接收$|D_i|\leqslant 3$.

以$\{EP_i\}_{i\in[1,3]}$为输入,S调用聚合加密多项式生成算法*PolyAgg*,得到聚合加密多项式$EP_a=\{7,SV\}=1x^1+4x^2+6x^3+7x^4+8x^5+9x^6+10x^7$. 同时,按照如下方式计算混淆的聚合解密多项式$DP^*=\{n,\boldsymbol{DV}^*\}$:

$$
\begin{aligned}
&DP^*\\
&=\textstyle\sum_{i=1}^{3}DP_i^*\\
&=\textstyle\sum_{i=1}^{3}(DP_i+\gamma_i)\\
&=\textstyle\sum_{i=1}^{3}DP_i+\sum_{i=1}^{3}\gamma_i\\
&=DP+\Gamma\\
&=(-x^2)+(-2x^3)+(-x^4)+(-x^5)+(-x^6)+\Gamma\\
&=\textstyle\sum_{i=1}^{7}(\boldsymbol{DV}[i]+\Gamma[i])x^i\\
&=\textstyle\sum_{i=1}^{7}\boldsymbol{DV}^*[i]x^i\\
&=\Gamma[1]x+(\Gamma[2]-1)x^2+(\Gamma[3]-2)+(\Gamma[4]-1)+(\Gamma[5]-1)x^5+(\Gamma[6]-1)x^6+\Gamma[7]x^7
\end{aligned}
$$

即

$\boldsymbol{DV}^*=[\Gamma[1],\Gamma[2]-1,\Gamma[3]-2,\Gamma[4]-1,\Gamma[5]-1,\Gamma[6]-1,\Gamma[7]]$.

之后,S选择随机数$r\in Z_p^*$并计算密文

$C_{10}=P(\Delta)_1^r=g^{\frac{rs_1}{(\alpha+1)(\alpha+4)(\alpha+6)}}$,

$C_{1j}=H(e(g^{\frac{1}{\alpha+j}},h)^r\|j)\cdot\boldsymbol{DV}^*[j]$,

$C_{20}=P(\Delta)_2^r=g^{\frac{rs_2}{(\alpha+2)(\alpha+3)(\alpha+6)}}$,

$C_{2j}=H(e(g^{\frac{1}{\alpha+j}},h)^r\|j)\cdot\boldsymbol{DV}^*[j]$,

$C_{30}=P(\Delta)_3^r=g^{\frac{rs_3}{(\alpha+1)(\alpha+3)(\alpha+5)}}$,

$C_{3j}=H(e(g^{\frac{1}{\alpha+j}},h)^r\|j)\cdot\boldsymbol{DV}^*[j]$,

其中,$1\leqslant j\leqslant 7$, $\boldsymbol{DV}^*[j]=\boldsymbol{DV}[j]+\Gamma[j]$. 最后,$S$将排序响应$C_i=\{C_{i0},C_{ij},EP_a\}$, $i\in[1,3]$, $j\in[1,7]$,发送给U_i.

客户端排序结果恢复实例. 当收到从云服务器S发来的排序响应$C_1=\{C_{10},C_{1j},EP_a\}$, U_1执行如下步骤恢复数据集$D_1=\{1,4,6\}$的排序结果$SR_1=\{sr_{11},sr_{12},sr_{13}\}$,其中$sr_{11},sr_{12},sr_{13}$分别代表数据1,4,6的排序结果. 首先,$U_1$将$EP_a$解析为$EP_a=\{7,\boldsymbol{SV}\}$,其中,$\boldsymbol{SV}=[1,4,6,7,8,9,10]$.

U_1通过C_{10},C_{1j},选择集合$\Delta_1=\{1,4,6\}$,私钥s_1和系统参数SP,计算数据集D_1对应的混淆后的解密向量$\boldsymbol{DV}^*$(由于1是全集当中最小的元素,其排序结果永远为1. 因此,U_1无需计算数据1的解密向量).

U_1解析$SR_1=\{sr_{11},sr_{12},sr_{13}\}$,可得数据1的排序结果为$sr_{11}=1$,数据4的排序结果$sr_{12}=\boldsymbol{SV}[4]+\boldsymbol{DV}^*[4]-\Gamma[4]=6$,数据6的排序结果$sr_{13}=\boldsymbol{SV}[6]+\boldsymbol{DV}^*[6]-\Gamma[6]=8$. $\boldsymbol{DV}^*[4]$和$\boldsymbol{DV}^*[6]$的计算见下.

同理,U_2,U_3可以恢复得到其数据集对应的排序结果.

$$
\begin{aligned}
\boldsymbol{DV}^*[4]&=C_{14}\cdot H(e(C_{10},h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+4)}})^{\frac{1}{s_1}}\|4)^{-1}\\
&=C_{14}\cdot H(e(P(\Delta)_1{}^r,h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+4)}})^{\frac{1}{s_1}}\|4)^{-1}\\
&=C_{14}\cdot H(e(g^{\frac{r\cdot s_1}{\prod_{j=1}^{3}(\alpha+d_{1j})}},h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+4)}})^{\frac{1}{s_1}}\|4)^{-1}\\
&=C_{14}\cdot H(e(g^{\frac{1}{\alpha+4}},h)^r\|4)^{-1}\\
&=H(e(g^{\frac{1}{\alpha+4}},h)^r\|4)\cdot\boldsymbol{DV}^*[4]\cdot\\
&\quad H(e(g^{\frac{1}{\alpha+4}},h)^r\|4)^{-1}\\
&=\Gamma[4]-1
\end{aligned}
$$

$$\begin{aligned}
\boldsymbol{DV}^*[4]&=C_{14}\cdot H(e(C_{10},h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+4)}})^{\frac{1}{s_1}}\|4)^{-1}\\
&=C_{14}\cdot H(e(P(\Delta)_1{}^r,h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+4)}})^{\frac{1}{s_1}}\|4)^{-1}\\
&=C_{14}\cdot H(e(g^{\frac{r\cdot s_1}{\prod_{j=1}^{3}(\alpha+d_{1j})}},h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+4)}})^{\frac{1}{s_1}}\|4)^{-1}\\
&=C_{14}\cdot H(e(g^{\frac{1}{\alpha+4}},h)^r\|4)^{-1}\\
&=H(e(g^{\frac{1}{\alpha+4}},h)^r\|4)\cdot\boldsymbol{DV}^*[4]\cdot\\
&\quad H(e(g^{\frac{1}{\alpha+4}},h)^r\|4)^{-1}\\
&=\Gamma[4]-1\\
\boldsymbol{DV}^*[6]&=C_{16}\cdot H(e(C_{10},h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+6)}})^{\frac{1}{s_1}}\|6)^{-1}\\
&=C_{16}\cdot H(e(P(\Delta)_1{}^r,h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+6)}})^{\frac{1}{s_1}}\|6)^{-1}\\
&=C_{16}\cdot H(e(g^{\frac{r\cdot s_1}{\prod_{j=1}^{3}(\alpha+d_{1j})}},h^{\frac{\prod_{j=1}^{3}(\alpha+d_{1j})}{(\alpha+6)}})^{\frac{1}{s_1}}\|6)^{-1}\\
&=C_{16}\cdot H(e(g^{\frac{1}{\alpha+6}},h)^r\|1)^{-1}\\
&=H(e(g^{\frac{1}{\alpha+6}},h)^r\|6)\cdot\boldsymbol{DV}^*[6]\cdot\\
&\quad H(e(g^{\frac{1}{\alpha+6}},h)^r\|6)^{-1}\\
&=\Gamma[6]-1
\end{aligned}$$

6　方案分析

本节首先分析协议Π_{PMMS}的正确性. 然后，我们进行安全性分析，证明协议Π_{PMMS}满足半诚实安全性和不合谋参与方穷举攻击下的恶意安全性.

6.1　正确性分析

在本文提出的协议Π_{PMMS}中，m个参与方分别拥有数据集$D_1,D_2,\cdots,D_m$. 协议Π_{PMMS}主要基于计数排序的思想来实现排序，即计算所有小于d_{ij}的数据的数量num_{ij}，从而得到d_{ij}的排序结果为$num_{ij}+1$. 协议的正确性证明如下：

定理1.　协议Π_{PMMS}正确地实现了功能$\mathcal{F}_{\text{PMMS}}$.

证明. 在协议Π_{PMMS}中，每个参与方U_i（$1\leqslant i\leqslant m$）首先通过*PolyEncode*算法将私有数据集D_i编码为多项式$P_i=\sum_{j=1}^{n}a_jx^j$. 其次，参与方U_i执行*PolyEnc*算法对多项式P_i进行加密. 该加密算法本质上是无条件安全的一次一密. 需要强调的是任一多项式P_i都可由最高次数n和系数向量$V_i=\{a_1,a_2,\cdots,a_n\}\in\{0,1\}^n$表示，即$P_i=\{n,V_i\}$. 因此，经过*PolyEnc*算法得到的长度为$n$比特的密钥向量为$\boldsymbol{BV}_i$、加密多项式为$EP_i=\{n,\boldsymbol{EV}_i\}$，其中$\boldsymbol{EV}_i[j]=V_i[j]\oplus\boldsymbol{BV}_i[j]$. 之后，利用解密多项式生成算法*PolyDec*计算出的解密多项式DP_i可以抵消由加密多项式EP_i造成的数据个数变化导致的排序结果的误差. 同样，其输出聚合解密多项式$DP=\{n,\boldsymbol{DV}\}$可以消除*PolyAgg*算法输出的聚合加密多项式$EP_a=\sum_{i=1}^{m}EP_i$带来的排序结果误差.

下面我们证明参与方U_i得到了数据集D_i中的数据d_{ij}的正确排序结果. 根据之前分析的计数排序的思想，数据d_{ij}的正确排序结果为

$$sr_{ij}=1+\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}\boldsymbol{V}_i[j].$$

在协议Π_{PMMS}中的排序结果恢复阶段，参与方U_i执行如下操作计算数据d_{ij}的排序结果：

$$\boldsymbol{SV}[d_{ij}]+\boldsymbol{DV}^*[d_{ij}]-\Gamma[d_{ij}].$$

因此，我们的目的就是证明：

$$sr_{ij}=1+\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}V_i[j]=\boldsymbol{SV}[d_{ij}]+\boldsymbol{DV}^*[d_{ij}]-\Gamma[d_{ij}].$$

首先，假设参与方U_i的聚合加密多项式、聚合解密多项式分别为EP_a和DP. 在EP_a，DP中第d_{ij}（$j>1$）次项$x^{d_{ij}}$的系数分别为$\boldsymbol{SV}[d_{ij}]$和$\boldsymbol{DV}[d_{ij}]$：

$$\boldsymbol{SV}[d_{ij}]=1+\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}\boldsymbol{EV}_i[j],$$

$$\boldsymbol{DV}[d_{ij}]=\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}(V_i[j]-\boldsymbol{EV}_i[j]).$$

另外，服务器计算得到的混淆聚合解密多项式为$DP^*=DP+\Gamma$. 则DP^*中第d_{ij}（$j>1$）次项$x^{d_{ij}}$的系数为$\boldsymbol{DV}^*[d_{ij}]=\boldsymbol{DV}[d_{ij}]+\Gamma[d_{ij}]$. 因此，我们有

$$\begin{aligned}
&\boldsymbol{SV}[d_{ij}]+\boldsymbol{DV}^*[d_{ij}]-\Gamma[d_{ij}]\\
&=\boldsymbol{SV}[d_{ij}]+(\boldsymbol{DV}[d_{ij}]+\Gamma[d_{ij}])-\Gamma[d_{ij}]\\
&=\boldsymbol{SV}[d_{ij}]+\boldsymbol{DV}[d_{ij}]\\
&=1+\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}\boldsymbol{EV}_i[j]+\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}(V_i[j]-\boldsymbol{EV}_i[j])\\
&=1+\sum_{i=1}^{m}\sum_{j=1}^{d_{ij}-1}\boldsymbol{V}_i[j]\\
&=sr_{ij}.
\end{aligned}$$

基于以上分析，我们证明了定理1，即所有参与方都能够得到他们所持有数据的正确排序结果.

证毕.

6.2　安全性分析

本节，我们证明协议Π_{PMMS}达到了半诚实安全性以及不合谋参与者穷举攻击下的恶意安全性. 根据定义1，需要为参与方U_i和服务器S构造模拟器Sim_i和Sim_s，使得Sim_i和Sim_s的输出分别与U_i和S的真实执行是不可区分的. 定理2中，我们在标准模型下基于模拟范式证明协议Π_{PMMS}达到了半诚实安全性. 根据定义2，我们给出定理3，并给出基于理想世界/现实世界范式的不合谋参与者穷举攻击下的恶意安全性证明.

定理2. 协议Π_{PMMS}针对半诚实敌手安全地实现了$\mathcal{F}_{PMMS}$.

证明. 下面分三种情况进行证明.

情况1：半诚实敌手A妥协了参与方U_1. 我们构造模拟器Sim_1来模拟敌手A妥协参与方U_1的真实执行. 由于敌手A是半诚实的，Sim_1可以直接获得U_1的真实输入$D_1=\{d_{11},d_{12},\cdots,d_{1k}\}$($|D_1|=k, d_{ij}\in[0,n]$)和输出$SR_1=\{sr_{11},sr_{12},\cdots,sr_{1k}\}$，并且将其发送给$\mathcal{F}_{PMMS}$. 当接收到来自敌手A的$T_1=\{P(\Delta)_1,\sum_1,k,EP_1,DP_1^*\}$，$Sim_1$执行以下步骤：

步骤1：基于输入T_1和系统参数SP，检查$e(P(\Delta)_1,\sum_1)=e(g,h^{\alpha^{n-k}})$是否成立. 如果等式不成立，则中止协议；否则接收$|D_1|\leqslant k$.

步骤2：选取随机数$\{r_1,r_2,\cdots,r_{m-1}\}\in\{0,1\}^n_{m-1}$. 令$\boldsymbol{EV}_i'=r_i$，生成模拟消息$\{\boldsymbol{EV}_i'\}_{i\in[1,m-1]}$；根据$EP_1=\{n,\boldsymbol{EV}_1\}$，以$\boldsymbol{EV}_1$和$\{\boldsymbol{EV}_i'\}_{i\in[1,m-1]}$为输入，调用聚合加密多项式生成算法$PolyAgg$生成聚合加密多项式$EP_a'=\{n,\boldsymbol{SV}'\}$.

步骤3：生成混淆解密多项式$DP^*=\{n,\boldsymbol{DV}^*\}$：生成随机多项式$\Gamma'=\sum_{j=1}^n\Gamma'[j]$. 对于$j\in D_i$，令$\boldsymbol{DV}^{\neq}[j]=sr_{ij}-SV'[j]+\Gamma'[j]$；否则，随机选取$rv\in[1,n]$，令$\boldsymbol{DV}^{\neq}[j]=rv$.

步骤4：选择随机数$r'\in Z_p^*$并计算密文$C_{0i}=P(\Delta)_i^{r'}=g^{\frac{r's_i}{(\alpha+d_{i1})(\alpha+d_{i2})\cdots(\alpha+d_{ik})}}$和$C_{ij}=H(e(g^{\frac{1}{\alpha+j}},h)\|j)^{r'}\cdot\boldsymbol{DV}^{\neq}[j]$，$1\leqslant i\leqslant m$，$1\leqslant j\leqslant n$.

最终，将排序响应$C_i=\{C_{0i},C_{ij},EP_a\}$，$i\in[1,m]$，$j\in[1,n]$返回给敌手A.

下面我们证明模拟器Sim_1的输出与U_1的真实执行过程是不可区分的.

首先，根据真实的协议执行，参与方U_1（敌手A）的view_1^{Π}为

$\text{view}_1^{PMMS}(D_1,\cdots,D_m)=(n,k,D_1,C_1=(C_{01},C_{1j})_{j\in[1,n]})$；

相反，模拟器Sim_1的输出为

$Sim_1(D_1,f_1(D_1,\cdots,D_m))=(n,k,D_1,C_1'=(C_{01}',C_{1j}')_{j\in[1,n]})$.

为了证明：

$Sim_1(D_1,f_1(D_1,\cdots,D_m))\overset{c}{\equiv}\text{view}_1^{PMMS}(D_1,\cdots,D_m)$，

我们证明：

$\{C_1'=(C_{01}',C_{1j}')_{j\in[1,n]}\}\overset{c}{\equiv}\{C_1=(C_{01},C_{1j})_{j\in[1,n]}\}$.

进一步，由于

$C_{01}'=P(\Delta)_1^r=g^{\frac{r'\cdot s_1}{(\alpha+d_{11})(\alpha+d_{12})\cdots(\alpha+d_{1k})}}$，

$C_{1j}'=H(e(g^{\frac{1}{\alpha+j}},h)\|j)^{r'}\cdot\boldsymbol{DV}^{\neq}[j]$，

$C_{01}=P(\Delta)_1^r=g^{\frac{r\cdot s_1}{(\alpha+d_{11})(\alpha+d_{12})\cdots(\alpha+d_{1k})}}$，

$C_{1j}=H(e(g^{\frac{1}{\alpha+j}},h)\|j)^{r}\cdot\boldsymbol{DV}^{*}[j]$.

我们需要证明：

$$C_{01}'\overset{c}{\equiv}C_{01},$$
$$C_{1j}'\overset{c}{\equiv}C_{1j};$$

对于C_{1j}'和C_{1j}，我们分别将其分成两个子集，即：

$C_{1j}'=\{C_{1j}'\}_{j\in\Delta}\cup\{C_{1j}'\}_{j\notin\Delta}=\{C_{1j}'\}_{j\in D_1}\cup\{C_{1j}'\}_{j\notin D_1}$，

$C_{1j}=\{C_{1j}\}_{j\in\Delta}\cup\{C_{1j}\}_{j\notin\Delta}=\{C_{1j}\}_{j\in D_1}\cup\{C_{1j}\}_{j\notin D_1}$；

根据[28]中的定理3，我们有：

$C_{01c}'\overset{c}{\equiv}C_{01}$和$\{C_{1j}'\}_{j\notin D_1}\overset{c}{\equiv}\{C_{1j}\}_{j\notin D_1}$；

另外，根据$\boldsymbol{DV}^{\neq}[j]=sr_{ij}-\boldsymbol{SV}'[j]+\Gamma'[j]$和排序结果恢复算法，$U_1$得到与真实排序结果$SR_1$相同的排序结果集合$SR_1'$. 因此，我们有$\{C_{1j}'\}_{j\in D_1}\overset{c}{\equiv}\{C_{1j}\}_{j\in D_1}$. 所以，可得$C_{1j}'\overset{c}{\equiv}C_{1j}$.

我们定义$h_0,h_1,\cdots,h_n$如下：

$h_0=C_1=(C_{01},C_{11},C_{12},\cdots,C_{1n})$

$h_1=(C_{01}',C_{11},C_{12},\cdots,C_{1n})$

$h_2=(C_{01}',C_{11}',C_{12},\cdots,C_{1n})$

$\cdots$

$h_n=C_1'=(C_{01}',C_{11}',C_{12}',\cdots,C_{1n}')$

显然，由上述分析，我们可得$h_0\overset{c}{\equiv}h_n$，即$C_1'\overset{c}{\equiv}C_1$. 因此，我们可以推出：

$Sim_1(D_1,f_1(D_1,\cdots,D_m))\overset{c}{\equiv}\text{view}_1^{PMMS}(D_1,\cdots,D_m)$.

情况2：半诚实敌手A妥协了参与方U_2或，…，或U_m；这种情况下的证明与情况1几乎完全相同. 因此，这里忽略证明过程.

情况3：半诚实敌手A妥协了服务器S. 我们构造模拟器Sim_S来模拟敌手A妥协的服务器S的真实执行. 相对于情况1和2，情况3较为简单，因为服务器S并没有输入和输出. 因此，我们只需要模拟出敌手A接收到的消息即可.

首先，Sim_S构造m个数据集$\{DS_i'\}_{i\in[1,m]}$，其中，$DS_i'=\{d_{i1}',d_{i2}',\cdots,d_{ik}'\}$作为选择集合. 基于$DS_i'$和系统参数$SP$，$Sim_S$选择随机数$s_i'\in Z_p^*$，计算$P'(\Delta)_i$和$\sum_i'$：

$P'(\Delta)_i=g^{\frac{s_i'}{(\alpha+d_{i1}')(\alpha+d_{i2}')\cdots(\alpha+d_{ik}')}}$，

$\sum_i'=h^{\frac{(\alpha+d_{i1}')(\alpha+d_{i2}')\cdots(\alpha+d_{ik}')\alpha^{n-k}}{s_i'}}$.

得到$\{P'(\Delta)_i,\sum_i'\}_{i\in[1,m]}$.

步骤2：以$\{DS_i'\}_{i\in[1,m]}$和n为输入，Sim_S通过调用基本编码算法$PolyEncode$获得多项式$\{P_i'\}_{i\in[1,m]}$. 之后，以$\{P_i'\}_{i\in[1,m]}$为输入，调用多项式加密算法$PolyEnc$，并输出加密多项式$\{EP_i'\}_{i\in[1,m]}$.

步骤3：以P_i'的系数向量V_i'和EP_i'的系数向量EV_i'为输入，Sim_S调用解密多项式生成算法$PolyDec$得到解密多项式DP_i'，$i\in[1,m]$. 然后，Sim_S将r_i加入到DP_i'中，得到$DP_i^*=DP_i'+r_i$.

步骤4：Sim_S构造排序请求$T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*\}$，$i\in[1,m]$，并分别提交给敌手A.

下面我们证明Sim_S的输出与服务器S的真实执行是不可区分的.

首先，根据真实的协议执行，服务器S（敌手A）的view_S^{Π}为：

$\mathrm{view}_S^{PMMS}(D_1,\cdots,D_m)=(n,k,\{T_i\}_{i\in[1,m]})$；

相反，模拟器Sim_1的输出为

$\mathrm{Sim}_S(n,k)=(n,k,\{T_i'\}_{i\in[1,m]})$.

为了证明：

$\mathrm{Sim}_S(n,k)\overset{c}{\equiv}\mathrm{view}_S^{PMMS}(D_1,\cdots,D_m)$，

我们需要证明：

$(n,k,\{T_i'\}_{i\in[1,m]})\overset{c}{\equiv}(n,k,\{T_i\}_{i\in[1,m]})$

进一步，我们需要证明：

$\{T_1',T_2',\cdots,T_m'\}\overset{c}{\equiv}\{T_1,T_2,\cdots,T_m\}$；

因此，我们首先证明对于$i\in[1,m]$，有$T_i'\overset{c}{\equiv}T_i$，即$\{P(\Delta)',\sum_i',k,EP_i',DP_i^{\sharp}\}\overset{c}{\equiv}\{P(\Delta),\sum_i,k,EP_i,DP_i^*\}$. 我们定义$H_0,H_1,H_2,H_3$如下：

$H_0=T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*\}$，

$H_1=\{P(\Delta)_i',\sum_i',k,EP_i,DP_i^*\}$，

$H_2=\{P(\Delta)_i',\sum_i',k,EP_i',DP_i^*\}$，

$H_3=T_i'=\{P(\Delta)_i',\sum_i',k,EP_i',DP_i^{\sharp}\}$.

由[28]中定理3可得$H_0\overset{c}{\equiv}H_1$；我们设计的多项式加密算法$PolyEnc$是一个一次一密加密算法，因此可得$H_1\overset{c}{\equiv}H_2$；同时，由于多项式$\Gamma$与$\Gamma'$为随机多项式，$DP_i^{\sharp}$和$DP_i^*$是不可区分的，可得$H_2\overset{c}{\equiv}H_3$.

综上可得$H_0\overset{c}{\equiv}H_3$，即$T_i'\overset{c}{\equiv}T_i$. 因此，我们可以通过类似过程得到：

$\{T_1',T_2',\cdots,T_m'\}\overset{c}{\equiv}\{T_1,T_2,\cdots,T_m\}$；

因此，$\mathrm{Sim}_S(n,k)\overset{c}{\equiv}\mathrm{view}_S^{PMMS}(D_1,\cdots,D_m)$. 证毕.

定理3. 协议Π_{PMMS}针对不合谋恶意参与方的穷举攻击中止安全地实现了$\mathcal{F}_{\mathrm{PMMS}}$.

证明. 在协议执行过程中，参与方U_i可能试图通过伪造超出规定大小k的数据集来得到更多的隐私（除自己数据的排序结果以外）. 在参与方U_i执行穷举攻击时，协议Π_{PMMS}能检测U_i的恶意行为并中止方案的运行.

我们首先考虑恶意敌手A妥协参与方U_1的情况，为U_1构造模拟器Sim_1. 首先Sim_1根据安全参数1^{λ}选择一个双线性映射e'：$G'\times G'\to G_T'$、两个生成元$g',h'\in G'$. 然后随机选择$\alpha'\in Z_p^*$作为系统密钥，并计算$g_j'=g^{\frac{1}{\alpha'+j}}$，$h_j'=h^{(\alpha')^j}$，$j\in[1,n]$. 此外，选择一个哈希函数$H'$：$G_T\to\{0,1\}^l$. Sim_1与U_1协商n阶随机多项式$\Gamma'=\sum_{j=1}^n\Gamma'[j]$，$\Gamma'[j]$代表$n$阶多项式$\Gamma'$的第$j$项. 之后，$\mathrm{Sim}_1$与$U_1$以多项式$\Gamma'$为输入，联合执行加法秘密共享协议. 协议结束后，$U_1$获得多项式$\Gamma'$的一个加法份额，即多项式$\gamma_1'$，$\mathrm{Sim}_1$获得多项式$\Gamma'$的$m$-1个加法份额，即多项式$\gamma_2',\gamma_3',\cdots,\gamma_m'$，满足$\Gamma'=\sum_{i=1}^m\gamma_i'=\sum_{j=1}^n\Gamma'[j]$. 最终，$\mathrm{Sim}_1$将$\{e',g',h',g_1',g_2',\cdots,g_m',h_1',h_2',\cdots,h_n',H'\}$发送给$U_1$.

当接收到参与方U_1的排序请求$T_1=\{P(\Delta)_1,\sum_1,k,EP_1,DP_1^*)\}$和选择集$\Delta_1'$，$\mathrm{Sim}_1$通过$\gamma_2',\gamma_3',\cdots,\gamma_m'$将$DP_1^*$进一步解密为$DP_1$. 根据$U_1$的解密多项式$DP_1$，$\mathrm{Sim}_1$恢复出$U_1$的编码多项式$P_1$和对应的加密多项式$EP_1$，最终得到$U_1$的数据集$D_1'$. Sim_1首先验证$|D_1'|\leqslant k$是否成立. 若不成立，Sim_1中止协议，发送Abort消息给参与方U_1，并以U_1的输出作为输出；否则，Sim_1将D_1'发送到$\mathcal{F}_{\mathrm{PMMS}}$，并接收来自$\mathcal{F}_{\mathrm{PMMS}}$的输出，即$D_1'=\{d_{11}',d_{12}',\cdots,d_{1k}'\}$的排序结果$SR_1'=\{sr_{11}',sr_{12}',\cdots,sr_{1k}'\}$.

Sim_1生成集合$\{d_{i1}',d_{i2}',\cdots,d_{ik}'\}_{i\in[2,m]}$，按照协议步骤生成对应的加密多项式$\{EP_i'\}_{i\in[2,m]}$和解密多项式$\{DP_i'\}_{i\in[1,m]}$. 之后，$\mathrm{Sim}_1$根据$U_1$的$EP_1$和$DP_1^*$以及上述生成的加密多项式和解密多项式进一步计算聚合加密多项式和聚合解密多项式$EP'=\sum_1^m EP_i'$和$DP'=\sum_1^m DP_i'$. 同时，Sim_1选择随机数$r'\in Z_p^*$并计算密文$C_{01}'=P(\Delta)_1^{r'}=g^{\frac{r's_1}{(\alpha+d_{11}')(\alpha+d_{12}')\cdots(\alpha+d_{1k}')}}$和$C_{1j}'=H(e(g^{\frac{1}{\alpha+j}},h)\|j)^{r'}\cdot(sr_{1j}'-DP'[j]+\Gamma[j])$，$j\in\Delta'$；$C_{1j}'=H(e(g^{\frac{1}{\alpha+j}},h)\|j)^{r'}\cdot(sr_j^*+DP'[j]+\Gamma[j])$，$j\notin\Delta'$，$sr_j^*$为随机数. Sim_1将排序响应$C_1'=\{C_{01}',C_{1j}',EP'\}$，$j\in[1,n]$，发送给$U_1$. 最终，$\mathrm{Sim}_1$输出所有模拟的消息.

我们对不合谋参与者进行穷举攻击时的安全性进行分析，并证明：$(\mathrm{IDEAL}_1^{\mathcal{F}}(D_1,f_i(D_1,D_2,\cdots,D_m))\overset{c}{\equiv}(\mathrm{REAL}_1^{\Pi}(D_1,D_2,\cdots,D_m))$，这里，$\Pi$代表协议$\Pi_{\mathrm{PMMS}}$，$\mathcal{F}$代表其功能$\mathcal{F}_{\mathrm{PMMS}}$.

情况1：恶意敌手A妥协参与方U_1，但是诚实地执行协议. 在U_1诚实执行协议的情况下，我们只需

要证明模拟器Sim_1与恶意敌手A的输出是不可区分的.

在理想世界中,Sim_1输出为:$(IDEAL^{\mathcal{F}}_1(D_1,f_i(D_1,D_2,\cdots,D_m))=(n,k,C'_1=(C'_{01},C'_{1j})_{j\in[1,n]},EP')$.

而在现实世界中,协议Π_{PMMS}是确定性的.因此,一旦所有参与方的输入$D_1\ D_2,\cdots,D_m$确定,Π_{PMMS}的输出也就确定了.其中,U_1的输出为:

$(REAL^{\Pi}_1(D_1,D_2,\cdots,D_m))=(n,k,C_1=(C_{01},C_{1j})_{j\in[1,n]},EP_a)$.根据Lai等人[28]的$n$选$k$的OT协议的安全性,$C'_{01}$和$C_{01}$是不可区分的.另外,$U_1$接收到$C'_1=(C'_{01},C'_{1j})_{j\in[1,n]}$与$C_1=(C_{01},C_{1j})_{j\in[1,n]}$,分别恢复出$D'_1$和$D_1$的排序结果$SR'_1$和$SR_1$,其中$D'_1=D_1$,因此,$SR'_1=SR_1$.因此,$C'_{1j}$与$C_{1j}$,$j\in\Delta'$,是不可区分的,而根据Lai等人的$n$选$k$的OT协议的安全性,$C'_{1j}$与$C_{1j}$,$j\notin\Delta'$也是不可区分的.

情况2:恶意敌手A妥协U_1,并伪造一个U_1的数据集$D'_1=\{d_1,d_2,\cdots,d_k\}$,$|D'_1|>k$.在此情况下,我们需要证明理想世界中的模拟器$Sim_1$和诚实参与方的输出与真实世界中的恶意敌手A和诚实参与方的输出应当是不可区分的.

在理想世界中,若恶意敌手A妥协了U_1,并伪造一个U_1的数据集$D'_1=\{d'_{11},d'_{12},\cdots,d'_{1k}\}$,$|D'_1|>k$,则$Sim_1$可以立刻根据其恢复的$D'_1$检测出来,并且向$U_1$发送Abort消息.因此,在理想世界中,恶意敌手和诚实参与方的联合分布为$\{n,k,Abort\}$.而在现实世界中,如果恶意敌手A妥协U_1并伪造数据集$D'_1=\{d'_{11},d'_{12},\cdots,d'_{1k}\}$,$|D'_1|>k$,那么,$U_1$根据$D'_1$构造排序请求$T_1=\{P(\Delta)_1,\sum_1,k,EP_1,E(DP^*_1)\}$,并将其发送至服务器$S$.根据排序请求$T_1$,服务器$S$通过验证等式$e(P(\Delta)_1,\sum_1)\stackrel{?}{=}e(g,h^{a^{n-k}})$立即检测出$|D|>k$,同时返回Abort消息.因此,在现实世界中,恶意敌手和诚实参与方的联合分布也为$\{n,k,Abort\}$.显然,恶意敌手和诚实参与方分别在理想世界和现实世界中的联合分布是不可区分的.

其次,我们考虑恶意敌手A妥协参与方$U_2,U_3,\cdots,U_m$的情况,模拟情况和U_1基本一致,在此忽略模拟及证明过程. 证毕.

7 性能评估与比较

本节我们对协议Π_{PMMS}的通信及计算开销进行评估,并同相关方案LDYW[25]和AHM+[27]进行比较.需要说明的是,虽然方案AHM+[27]是一个两方设置下的不经意排序协议,但在该方案中,“两方”实际上指的是两个服务器,分别表示为S_1和S_2.在该协议中,S_1和S_2分别持有待排序数据的秘密份额.以数据的秘密份额为输入,双方联合执行不经意排序协议以完成对数据的排序.因此,本实验通过以下方式来实现方案AHM+[27]在多参与方设置下的性能评估:在多方(假设为m方)设置下,首先将m个参与方数据集的秘密共享份额分配给两个服务器S_1和S_2.之后,以所有数据的秘密份额为输入,S_1和S_2执行不经意排序,从而完成对多参与方数据集的不经意排序.

实验中的代码均使用C++实现,实验环境为Inter(R) Core(TM) i7-12700F CPU,Windows 10系统及16GB RAM.由于协议的计算和通信开销与参与方数量m,参与方持有数据的数量k及数据范围n密切相关,我们分别在以下三种设置下评估协议Π_{PMMS}与相关协议的性能表现:

设置1:数据量k取值{8000,10 000,12 000,14 000,16 000,18 000,20 000},参与方数量m和数据域上界n分别取15和500 000;

设置2:参与方数量m取值{10,15,20,25,30,35,40},数据域上界n和数据量k分别取500 000和20 000;

设置3:数据范围n取值{200 000,250 000,300 000,350 000,400 000,450 000,500 000},数据量k和参与方数量m分别取20 000和15.

此外,为了方便对比,我们使用下面这些符号:l_G表示群G中元素的大小;$|E_G|$表示ElGamal加密密文E_G的大小;l表示数据集中数据的比特长度.

7.1 通信开销比较

本节,我们在上述三种设置下对协议Π_{PMMS}的通信开销进行评估,并同相关工作LDYW[25]和AHM+[27]进行比较.

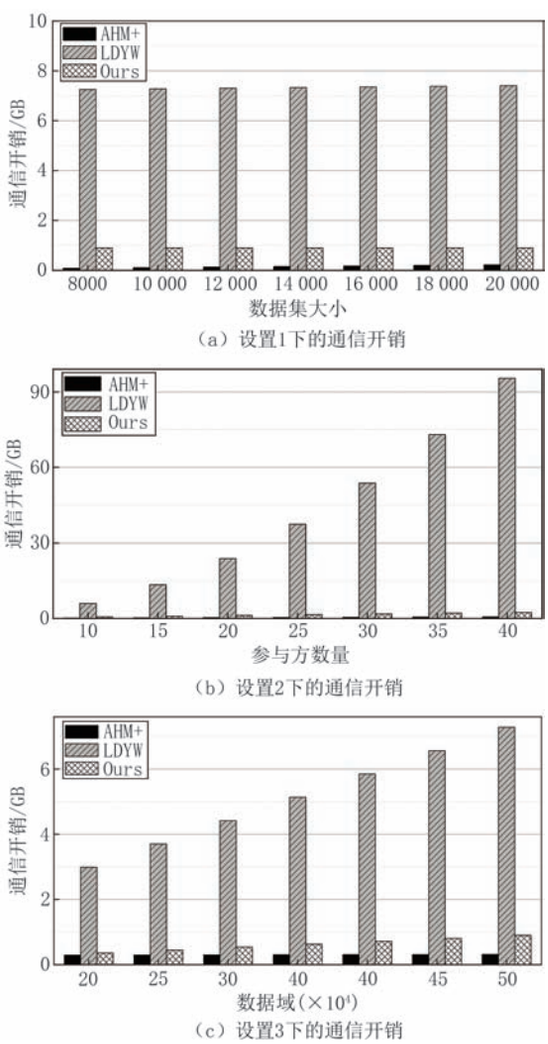
在协议Π_{PMMS}中,参与方之间并不直接进行通信,而只与服务器通信.每个参与方和云服务器之间的通信开销包括两部分:一是参与方U_i向服务器发送排序请求$T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP^*_i\}$;二是服务器向参与方$U_i$发送排序结果$C_i=\{C_{0i},C_{ij},EP_a\}$,$j\in[1,n]$.因此,在协议$\Pi_{PMMS}$中,每个参与方的通信复杂度为$O(n)$,具体通信开销为$|P(\Delta)_i,\sum_i,k,EP_i,DP^*_i|+|C_{0i},C_{ij},EP_a|_{j\in[1,n]}=(n+3)l_G+n\ \log m+n+kl$.因此,协议$\Pi_{PMMS}$的总通信开销为:$m((n+3)l_G+n\log m+n+kl)$.在协议LDYW[25]中,每个参与方的通信开销包含两部分:一是参与方U_i发送给其他$m-1$个参与方的大小为n的密文向量$\{E_{G1},E_{G2},\cdots,$

$E_{Gn}\}$. 二是参与方U_i与其他参与方协同对排序结果密文进行解密，得到排序结果明文，通信开销为$k\times(m-1)$个密文. 因此，在该协议中，每个参与方的通信复杂度为$O(n(m-1)+k(m-1))$，通信开销为$(nm+km-n-k))|E_G|$. 因此，该方案的总通信开销为：$m(nm+km-n-k))|E_G|$. 而在AHM+[27]中，通信开销主要来自各服务器之间的秘密共享，其通信复杂度为$O(\frac{kl}{L}(2^L+1)-\lambda k+2lk-Lk)$，具体通信开销为$3kl+6k(\frac{l}{L}(2^L+1)\lambda-6\lambda+3l-2L))$，其中，$L=3$. 因此，总通信开销为：$3mkl+6mk(\frac{l}{L}(2^L+1)\lambda-6\lambda+3l-2L))$. 表2总结了各协议的通信开销和通信轮次复杂度.

表2　通信开销比较

协议	通信开销	通信轮次复杂度
AHM+[27]	$3mkl+6mk(\frac{l}{L}(2^L+1)\lambda-6\lambda+3l-2L))$	$O(l/L)$
LDYW[25]	$m(nm+km-n-k))\|E_G\|$	$O(m)$
Ours	$m((n+3)l_G+n\log m+n+kl)$	$O(1)$

针对设置1，我们分别计算了协议Π_{PMMS}、LDYW[25]和AHM+[27]具体的通信开销. 为了更直观地进行比较，图5(a)展示了本文协议Π_{PMMS}、LDYW[25]和AHM+[27]随着数据量k从8000增长到20 000时的通信开销变化. 从图可以看到，LDYW[25]的通信开销最大，AHM+[27]的通信开销最小，而协议Π_{PMMS}的通信开销介于二者之间. 具体来说，当数据集大小为20 000时，Π_{PMMS}的通信开销是898.44 MB，为协议LDYW[25]通信开销的12.08%，大约是AHM+[27]的4倍. 原因在于，在协议Π_{PMMS}中，参与方发送的排序请求中仅仅包含5条消息，但是为了实现不合谋参与方穷举攻击下的协议安全性，参与方在排序响应阶段需要接收一个聚合的多项式和$n+1$个群Z_p中的元素，从而增加了协议Π_{PMMS}的通信开销. 然而，尽管协议LDYW[25]仅仅保证了半诚实安全，无需对参与方的恶意穷举行为进行防御从而节省了大量通信开销，但每个参与方仍然需要发送$(m-1)n$条ElGamal密文消息，并且需要和其他$m-1$个参与方进行通信从而实现密文的联合解密. 同样，协议AHM+[27]也无需进行额外的通信验证穷举攻击，各参与方仅需对各自拥有的每个数据进行秘密共享；尽管各个服务器之间需要进行多轮迭代从而得到最终的排序结果，但在上述迭代过程中不需要传输密文，而只需发送数据的份额. 因此，具有较低的通信开销. 同时，需要注意的是，在设置1下，协议Π_{PMMS}和协议LDYW[25]的通信开销几乎是常量，这是因为在协议Π_{PMMS}和协议LDYW[25]中，参与方的通信开销独立于数据量k，而AHM+[27]的通信开销则随着k的增大呈线性增长. 在设置1下，协议Π_{PMMS}的通信效率显著高于LDYW[25]，低于AHM+[27]. 总体而言，我们的协议Π_{PMMS}的通信效率是可接受的.



(a) 设置1下的通信开销

(b) 设置2下的通信开销

(c) 设置3下的通信开销

图5　通信开销对比

针对设置2，图5(b)刻画了Π_{PMMS}、LDYW[25]和AHM+[27]随着参与方数量m从10增加到40时的通信开销变化. 从图我们可以看出，随着参与方数量的增多，本文协议Π_{PMMS}与AHM+[27]的通信开销远低于LDYW[25]且增长都十分缓慢，而LDYW[25]通信开销非常大且呈线性增长. 因此，Π_{PMMS}在m增长这

一条件下的通信是非常高效的.具体来说,当参与方数量为40时,本文协议Π_{PMMS}的通信开销为2453.36 MB,分别约为方案LDYW[25]和方案AHM+[27]通信开销的2.51%和3.53倍.在Π_{PMMS}中,当参与方数量m增大,各个参与方的通信量略有增加.然而这些增长与总体的通信开销相比较小.因此,Π_{PMMS}适合参与方数量较大的多方多数据排序场景.而在协议LDYW[25]中,当参与方数量m增大,每个参与方需要额外增加$O(n)$个密文和$O(k)$共享解密密文,其中,n的值相对较大.因此,随着参与方数量m增大,各参与方通信开销增长显著.总之,本文协议Π_{PMMS}在通信效率方面远远高于LDYW[25],而略低于AHM+[27].因此,本文协议Π_{PMMS}具有高效的通信,并且适用于多参与方的场景.

针对设置3,我们在数据域上界n变化这一条件下对协议Π_{PMMS}、LDYW[25]和AHM+[27]的通信开销进行比较.图5(c)展示了协议Π_{PMMS}、LDYW[25]和AHM+[27]之间的通信开销对比结果.从图可以看出,随着数据域上界不断变大,协议LDYW[25]通信开销较大且线性增长,协议AHM+[27]具有较小的通信开销,而协议Π_{PMMS}的通信开销远远小于协议LDYW[25],接近于协议AHM+[27].具体来说,当数据域上界n=500 000时,协议Π_{PMMS}的通信开销为919.11MB,约为LDYW[25]的12.33%,AHM+[27]的286.87%.在协议Π_{PMMS}中,为了保证排序结果的隐私性,采用i-OT协议保护n个解密因子(实际上仅使用了其中的k个)的隐私.具体来讲,从G_T中生成$n+1$个元素来加密n个解密因子,其通信复杂度为$O(n)$.因此,通信开销随着n不断增加呈线性增长.然而在LDYW[25]中,各参与方合作构建一个大小为$m\cdot n$的密文矩阵,并且每个参与方都会收到来自其他参与方的密文消息构建的一个密文矩阵,故其通信复杂度为$O(m\cdot n)$.因此,随着n的不断增长,Π_{PMMS}的通信开销相对于LDYW[25]增长较慢.与协议AHM+[27]相比,当n较小时,Π_{PMMS}的通信开销几乎与AHM+[27]相当;然而当n较大时,AHM+[27]的通信开销要小于Π_{PMMS}.这是由于本文协议Π_{PMMS}在实现了半诚实安全基础上还能防御不合谋参与方的恶意窃举攻击行为,从而导致本文协议通信开销的增加.

基于以上的比较和分析,在以上三种设置下,相较于LDYW[25]和AHM+[27],我们的协议Π_{PMMS}总体上具有合理的通信效率.

7.2 计算开销比较

本节,我们首先对本文协议Π_{PMMS}的在线计算开销进行分析.其次,我们在三种不同实验设置下评估协议Π_{PMMS}的在线计算开销,并与LDYW[25]和AHM+[27]的在线计算开销进行比较.

协议Π_{PMMS}的计算开销主要分为三部分:(1)参与方计算排序请求;(2)服务器计算排序结果;(3)参与方恢复排序结果.我们依次对这三部分计算开销进行分析.

在第(1)部分中,每个参与方U_i将数据集D_i作为选择集合Δ_i,即$\Delta_i=D_i$,并构造排序请求$T_i=\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*)\}$,其中,$P(\Delta)_i$的计算需要$k$次幂指数运算和$k-1$次乘法运算,$\sum_i$的计算同样需要$k$次幂指数运算和$k-1$次乘法运算.而$EP_i,DP_i^*$的计算则仅涉及异或运算和其他的简单数学运算.因此,这一部分的计算开销主要来自于计算$P(\Delta)_i$和$\sum_i$,$1\leqslant i\leqslant m$,且计算次数主要与参与方数量m和k有关.综合来说,参与方U_i主要需要$2k$次幂指数运算和$2k-2$次乘法运算来生成排序请求.

在第(2)部分,基于系统参数SP和收到的排序请求$T_{i=}\{P(\Delta)_i,\sum_i,k,EP_i,DP_i^*\}$,$1\leqslant i\leqslant m$,服务器$S$为每个参与方计算两次相对耗时的双线性映射$e(P(\Delta_i),\sum_i)$和$e(g,h_{n-k}=h^{a^n-k})$,并根据计算结果判断$e(P(\Delta),\sum_i)\stackrel{?}{=}e(g,h^{a^n-k})$是否成立.由于$e(g,h_{n-k}=h^{a^n-k})$的计算仅依赖于公开系统参数$SP$,服务器$S$可以通过预计算来避免这些计算开销,即服务器$S$只需计算$m$次而非$2m$次双线性配对操作.另外,$S$直接计算聚合加密多项式$EP_a$和混淆聚合解密多项式$DP^*$,$DP^*=DP+\Gamma$.这里的计算仅涉及高效的多项式加法,计算开销较小.随后,S根据混淆聚合解密多项式DP^*的系数向量$\boldsymbol{DV}^*$计算密文$C_{0i}=P(\Delta_i)^r$,$C_{ij}=H(e(g^{\frac{1}{\alpha+j}},h)^r)\cdot\boldsymbol{DV}^*[j]$,$j\in[1,n]$,$i\in[1,m]$.$C_{0i}$的计算涉及$m$个幂指数运算,而$C_{ij}$的计算则涉及$m\cdot n$个双线性映射和哈希操作.虽然参与方数量$m$相对较小,但是数据的取值上界$n$相对较大.因此,$m\cdot n$非常大.所以,这里$C_{ij}$的计算是本部分乃至整个协议的计算开销瓶颈.然而,值得注意的是,$H(e(g^{\frac{1}{\alpha+j}},h)^r$的计算独立于服务器$S$接收的来自参与方的排序请求,而仅与公开系统参数$SP$和服务器生成的随机数$r$有关.因此,$S$可以通过预计算来避免这一巨大的开销,从而大幅提升协议的计算效率.综合来说,服务器S需要m次双线性映射操作、

m次幂指数运算操作以及n次乘法操作为所有用户计算排序结果.

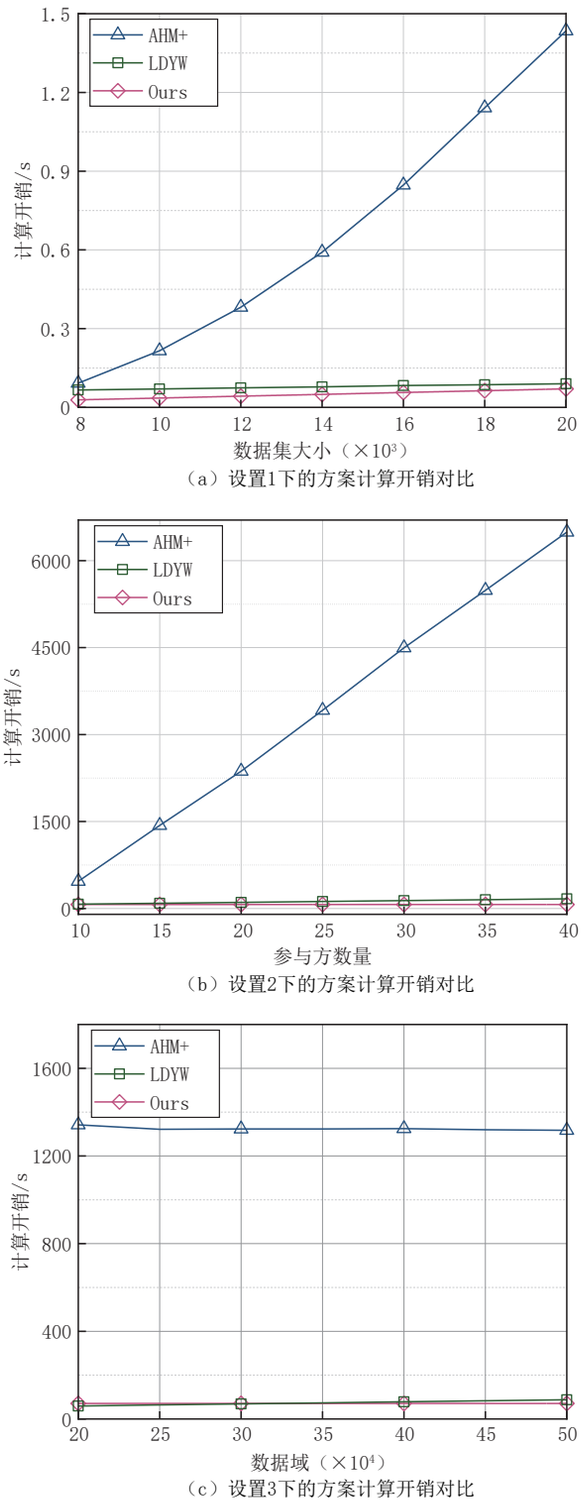
在第(3)部分,参与方U_i收到来自服务器S的排序响应,通过C_{0i},C_{ij},选择集合Δ_i,私钥s_i和系统参数SP,计算混淆后的解密向量DV^*,其中$DV^*[j]=C_{ij}\cdot H(e(C_{0i},h^{\frac{\sum_j^k(\alpha+d_{ij})}{(\alpha+d_{ij})}\frac{1}{s_i}})^{-1}$,$j\in\Delta_i$,$|\Delta_i|=k$. 具体来说,$U_i$需要计算$k$次双线性映射和哈希操作. 另外,$U_i$通过简单高效的数学运算计算$sr_{ij}$. 因此,在这一部分,$U_i$的主要计算开销为$k$次双线性映射和$k$次哈希操作.

同时,我们分别在上述三种设置下评估了Π_{PMMS}、LDYW[25]和AHM+[27]的总体计算开销,并将这三种条件设置下的计算开销实验结果分别展示在图6(a)、图6(b)和图6(c)中.

图6(a)描述了Π_{PMMS}、LDYW[25]和AHM+[27]总体计算开销随着数据集大小k从8000增加到20 000时的变化趋势. 从图中我们可以发现,随着数据集大小k不断增大,协议Π_{PMMS}、LDYW[25]和AHM+[27]的计算开销都呈增长趋势. 具体来说,随着k的增大,Π_{PMMS}的总体计算开销始终小于协议LDYW[25]和AHM+[27]的计算开销. 其中,LDYW[25]的计算开销要高于Π_{PMMS}的计算开销,而AHM+[27]的计算开销最大,并且远远高于Π_{PMMS}. 这是由于协议AHM+[27]需要对每一个数据进行秘密共享,因此其计算开销会随着数据量的增加而显著增长. 而对于协议Π_{PMMS}和协议LDYW[25]来说,数据量的增长仅仅影响到排序结果恢复阶段的计算开销. 具体来说,协议LDYW[25]的排序结果恢复需要执行与数据量k成正比的联合解密,而协议Π_{PMMS}则需要执行正比于数据量k的双线性映射. 因此,协议Π_{PMMS}与LDYW[25]的计算开销都具有缓慢的增长趋势.

图6(b)显示了Π_{PMMS}、LDYW[25]和AHM+[27]在参与者数量m从10增长到40时的计算开销变化情况. 由图可知,随着参与者数量m的增加,本文协议Π_{PMMS}始终具有最低的计算开销. 具体来说,协议Π_{PMMS}、LDYW[25]和AHM+[27]的总体计算开销均呈增长趋势. 其中,Π_{PMMS}的计算开销和计算开销的增长速率都要小于LDYW[25],而远远小于AHM+[27]. 主要原因是在协议AHM+[27]中,参与方数量m的增加导致服务器需共享的数据量增加. 并且,数据秘密共享的过程无法并行. 因此,协议AHM+[27]的计算开销会随着参与方的增多而显著增长. 而对于协议LDYW[25]来说,m的增大会导致

总体的密文矩阵规模增大,从而使得每个参与方在恢复排序结果时需执行更多的密文同态计算. 因此,其计算开销具有较小的增长趋势. 而协议Π_{PMMS}的计算开销几乎独立于参与方的数量,因而随着参与方数量的增长,其计算开销几乎保持恒定.



（a）设置1下的方案计算开销对比

（b）设置2下的方案计算开销对比

（c）设置3下的方案计算开销对比

图6　计算开销对比

图6(c)展示了随着数据域n从200 000变化到500 000各方案计算开销的变化情况. 显然,随着数

据域上界n不断增大，三个协议均具有几乎恒定的计算开销．其中，协议Π_{PMMS}和协议LDYW[25]的计算开销十分接近，且均显著优于协议AHM+[27]的计算开销．这是因为协议AHM+[27]的计算开销受数据量k的影响很大，而几乎不受数据域上限n的影响．然而，协议Π_{PMMS}和LDYW[25]均能执行大量的预计算，从而使得计算开销独立于n.

通过上述分析，我们可以得出结论，相比与方案LDYW[25]和AHM+[27]，本文协议Π_{PMMS}在三种设置下都具有高效的计算．具体来说，当数据集大小为20 000，参与方数量为15，数据域为500 000时，协议Π_{PMMS}的计算开销约为69.76s，分别约为LDYW[25]和AHM+[27]的76.85%和4.86%.

8 结论及展望

基于所采用的基于多项式的编码方法，本文提出了一个高效的隐私保护多方多数据排序协议Π_{PMMS}，该协议允许多个参与方以隐私保护的方式对其拥有的多个数据进行排序．我们采用了经典的半诚实安全定义，同时给出了针对恶意穷举攻击的中止安全定义．形式化的安全性分析证明了Π_{PMMS}在经典半诚实模型和我们定义的穷举攻击恶意模型下是安全的．此外，我们通过大量的实验对协议Π_{PMMS}和其他相关协议的性能表现进行了评估．评估结果表明Π_{PMMS}是通信和计算高效的．总的来说，本文所提出隐私保护多方多数据排序方案能够达到多参与方、多数据、隐私保护和高效率等多层面的预期设计目标．未来，我们将研究不同应用场景（如安全拍卖）下更高效的、隐私保护的非去重/去重排序方案．

参 考 文 献

[1] Lindell Y. Secure multiparty computation. Communications of the ACM, 2020, 64(1): 86-96

[2] Yao A C. Protocols for secure computations//Proceedings of the IEEE Symposium on Foundations of Computer Science. Chicago, USA, 1982: 160-164

[3] Dong C, Chen L, Wen Z. When private set intersection meets big data: an efficient and scalable protocol//Proceedings of the ACM SIGSAC Conference on Computer &Communications Security. New York, USA, 2013: 789-800

[4] Pinkas B, Schneider T, Zohner M. Faster private set intersection based on OT extension//Proceedings of the USENIX Security Symposium. San Diego, USA, 2014: 797-812

[5] Chen H, Laine K, Rindal P. Fast private set intersection from homomorphic encryption//Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Dallas, USA, 2017: 1243-1255

[6] Pinkas B, Schneider T, Zohner M. Scalable private set intersection based on OT extension. ACM Transactions on Privacy and Security, 2018, 21(2): 1-35

[7] Chor B, Kushilevitz E, Goldreich O, et al. Private information retrieval. Journal of the ACM, 1998, 45(6): 965-981.

[8] Goldberg I. Improving the robustness of private information retrieval//Proceedings of the IEEE Symposium on Security and Privacy. Oakland, USA, 2007: 131-148

[9] Sun H, Jafar S A. The capacity of robust private information retrieval with colluding databases. IEEE Transactions on Information Theory, 2017, 64(4): 2361-2370

[10] Naor M, Pinkas B. Oblivious transfer and polynomial evaluation// Proceedings of the ACM symposium on Theory of Computing. Atlanta, USA, 1999: 245-254

[11] Döttling N, Garg S, Hajiabadi M, et al. Two - round oblivious transfer from cdh or lpn//Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques. Zagreb, Croatia, 2020: 768-797

[12] Choudhuri A R, Ciampi M, Goyal V, et al. Oblivious transfer from trapdoor permutations in minimal rounds//Proceedings of the Theory of Cryptography Conference. Raleigh, USA, 2021: 518-549

[13] Jönsson K V, Kreitz G, Uddin M. Secure multi - party sorting and applications. Cryptology ePrint Archive, 2011, 2011(122): 1-19

[14] Hamada K, Kikuchi R, Ikarashi D, et al. Practically efficient multipa-rty sorting protocols from comparison sort algorithms// Proceedings of the International Conference on Information Security and Cryptology. Seoul, Korea, 2012: 202-216

[15] Dehghan M, Sadeghiyan B. Secure multi-party sorting protocol based on distributed oblivious transfer//Proceedings of the 10th International Conference on Computer and Knowledge Engineering. Mashhad, Iran, 2020: 011-017

[16] Ajtai M, Komlós J, Szemerédi E. An o ($n\log n$) sorting network//Proceedings of the ACM Symposium on Theory of Computing. Boston, USA, 1983: 1-9

[17] Goldreich O, Micali S, Wigderson A. How to play any mental game//Proceedings of the ACM symposium on Theory of computing. New York, USA, 1987: 218-229

[18] Liu Wen, Luo Shou-Shan, Chen Ping. Solution of secure multi-party multi-data raking problem based on El Gamal encryption. Journal on Communications, 2007, 28(11): 1-5 (in Chinese)
(刘文，罗守山，陈萍．利用ElGamal密码体制解决安全多方多数据排序问题．通信学报，2007，28(11)：1-5)

[19] Xiao Qian, Luo Shou-Shan, Chen Ping, et al. Research on the problem of secure multi-party ranking under semi-honest model. Acta Electronica Sinica, 2008,36(04): 709-714 (in Chinese)
(肖倩，罗守山，陈萍等．半诚实模型下安全多方排序问题的研究．电子学报，2008，36(04)：709-714)

[20] Qiu Mei, Luo Shou-Shan, Liu Wen, et al. A solution of secure multi-party multi-data ranking problem based on RSA encryption scheme. Acta Electronica Sinica, 2009, 37(5): 1119-1123 (in Chinese)
(邱梅,罗守山,刘文等.利用RSA密码体制解决安全多方多数据排序问题.电子学报,2009,37(05):1119-1123)

[21] Bogdanov D, Laur S, Talviste R. A practical analysis of oblivious sorting algorithms for secure multi-party computation//Proceedings of the Nordic Conference on Secure IT Systems. Tromsø, Norway, 2014: 59-74

[22] Li Shun-Dong, Kang Jia, Yang Xiao-Yi, et al. Secure multiparty characters sorting. Chinese Journal of Computers, 2018, 41(5): 1172-1188 (in Chinese)
(李顺东,亢佳,杨晓艺等.多个字符排序的安全多方计算.计算机学报,2018,41(05):1172-1188)

[23] Wang Ning, Gu Hao-Min, Zheng Tong. A practical and efficient secure multi-party sort protocol. Computer Applications and Software, 2018, 35(10): 305-311 (in Chinese)
(王宁,顾昊旻,郑彤.一种实用高效的安全多方排序协议.计算机应用与软件,2018,35(10):305-311)

[24] Dai, H, Ren, H, Chen, Z, et al. Privacy-preserving sorting algorithms based on logistic map for clouds. Security and Communication Networks, 2018, 2018: 1-10

[25] Li Shun-Dong, Du Run-Meng, Yang Yan-Jing, et al. Secure multiparty multi-data ranking. Chinese Journal of Computers, 2020, 43(08): 1448-1462 (in Chinese)
(李顺东,杜润萌,杨颜璟等.安全多方多数据排序.计算机学报, 2020, 43(8): 1448 - 1462)

[26] Chida, K, Hamada, K, Ikarashi, D, et al. An efficient secure three-party sorting protocol with an honest majority. Cryptology ePrint Archive, 2019

[27] Attrapadung N, Hanaoaka G, Matsuda T, et al. Oblivious linear group actions and applications//Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Korea, 2021: 630-650

[28] Lai J, Mu Y, Guo F, et al. Efficient k-out-of-n oblivious transfer scheme with the ideal communication cost. Theoretical Computer Science, 2018, 714: 15 - 26

[29] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612-613

[30] Delerablée C, Paillier P, Pointcheval D. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys//Proceedings of the International Conference on Pairing-Based Cryptography. Tokyo, Japan, 2007: 39-59

[31] Goldreich O. Foundations of cryptography: Basic applications. Cambridge, Cambridge university press, 2009

[32] Chase M, Miao P. Private set intersection in the internet setting from lightweight oblivious PRF//Proceedings of the International Cryptology Conference, Santa Barbara, USA, 2020: 34-63

SHANG Shuai, Ph.D. candidate. His research interests include data security and privacy computing.

LI Xiong, Ph. D., professor, Ph. D. supervisor. His research interests include data security and privacy computing.

ZHANG Wen-Qi, M.S. candidate. His research interests include data security and privacy computing.

WANG Xiao-Fen, Ph. D., associate professor. Her research interests include applied cryptography and security protocols.

LI Zhe-Tao, Ph. D., professor, Ph. D. supervisor. His research interests include computer networks, artificial intelligence and security.

ZHANG Xiao-Song, Ph. D., professor, Ph. D. supervisor. His research interests are blockchain, AI security, etc.

Background

Secure Multi-Party Computation (MPC) is a technology that uses cryptography to enable multiple participants to calculate and analyze these data without exposing their private data. With the increasing demand for data privacy protection, the application of secure multi-party computing is becoming more and more extensive. In some scenarios, multiple participants need to sort their own data, but due to factors such as data privacy, the data cannot be exposed to other parties. As a solution to this problem, secure multi-party sorting has also received extensive attention and application. As an application of MPC, secure multi-party sorting can realize the requirement of sorting multi-party data in a privacy-preserving manner. At present, secure multi-party sorting has been widely used in finance, medical care, e-commerce and other fields to protect the data privacy of all parties. At the same time, related research is also deepening, and many new algorithms and technologies have emerged, such as the secure multi-party sorting algorithm based on deep learning, and the secure multi-party sorting algorithm based on homomorphic encryption.

Secure multi-party sorting has a wide variety of applications in reality, which puts forward new requirements for secure multi-party sorting, including the functional requirements and security requirements. In terms of the functionality, more and more

applications involve more parties with large data sets. Thus, the secure multi-party sorting protocol which supports multiple parties, each with a large data set is in badly need. As for the security features, the secure multi-party sorting protocol which achieves malicious security or considers the malicious attacks of the participants are really eagerly awaited.

However, the existing work of secure multi-party sorting is far from satisfactory. Most of the protocols achieve secure multi-party multi-data at the expense of low communication and computation efficiency. Moreover, none of the researches focus on the active attack behaviors of the participants, e. g., the exhaustive attacks. Aim to address the above challenges, this work not only realizes the secure multi-party multi-data sorting via our proposed lightweight polynomial-based encoding algorithms but also take into account the malicious exhaustive attacks of the participants through an oblivious transfer technique. Finally, we conclude the paper and highlight future challenges.

It can be predicted that with the continuous development and improvement of technology, secure multi-party sorting will be more practical and widely used in more fields.

This work was supported by the National Natural Science Foundation of China (Key Program 62332018, General Programs 62072078 and 62271128), the National Natural Science Foundation of Sichuan Province (2022NSFSC0550), and the Open Project of the Key Laboratory of Data Protection and Intelligent Management, Ministry of Education, Sichuan University (SCUSAKFKT202303Z). All of these projects are related to data security and privacy protection.