计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.20220514
59(10):2338-2347,2022

基于秘密分享的高效隐私保护四方机器学习方案

阎允雪[1] 马 铭[1] 蒋 瀚[1,2]
[1](山东大学软件学院 济南 250101)
[2](山东省软件工程重点实验室(山东大学) 济南 250101)
(202020645@mail.sdu.edu.cn)

An Efficient Privacy Preserving 4PC Machine Learning Scheme Based on Secret Sharing

Yan Yunxue[1], Ma Ming[1], and Jiang Han[1,2]
[1](*School of Software*, *Shandong University*, *Jinan* 250101)
[2](*Key Laboratory of Software Engineering of Shandong Province* (*Shandong University*), *Jinan* 250101)

Abstract The wide application of machine learning technology makes user data face a serious risk of privacy leakage, and the privacy-preserving distributed machine learning protocol based on secure multi-party computation technology has become a widely concerned research field. In order to obtain a more efficient protocol, Chaudhari et al. proposed the Trident quadrilateral protocol framework. On the basis of the tripartite protocol, an honest participant is introduced as a trusted third party to execute the protocol, and the Swift framework proposed by Koti et al. is to select an honest participant as a trusted third party to complete the protocol through a screening process under the background of a three-party protocol with honest majority of participants. The framework to an honest-majority quadrilateral protocol is generalized. Under such a computing framework, a trusted third party obtains sensitive data of all users, which violates the original intention of secure multi-party computation. To solve this problem, a four-party machine learning protocol based on (2,4) secret sharing is designed. By improving the honest party screening process of the Swift framework, two honest parties can be determined and a semi-honest secure two-party computing protocol which can efficiently complete computing tasks is executed. The protocol transfers 25% of the communication load from the online phase to the offline phase, which improves the efficiency of the online phase of the scheme.

Key words secure multi-party computation; privacy preserving; machine learning; secret sharing; malicious adversaries

摘 要 机器学习技术的广泛应用使得用户数据面临严重的隐私泄露风险,而基于安全多方计算技术的隐私保护分布式机器学习协议成为广受关注的研究领域.传统的安全多方计算协议为了实现恶意敌手模型下的安全性,需要使用认证秘密分享、零知识证明等工具,使得协议实现效率较低.为了得到更高效的协议,Chaudhari 等人提出 Trident 四方协议框架,在三方协议的基础上,引入一个诚实参与方作为

收稿日期:2022-06-11;**修回日期**:2022-08-11
基金项目:国家自然科学基金项目(62172258);山东省软件工程重点实验室科技创新基地专项(11480004042015)
This work was supported by the National Natural Science Foundation of China (62172258) and the Special Project of Science and Technology Innovation Base of Key Laboratory of Software Engineering of Shandong Province (11480004042015).
通信作者:蒋瀚(jianghan@sdu.edu.cn)

可信第三方来执行协议;而 Koti 等人提出的 Swift 框架,在参与方诚实大多数的三方协议背景下,通过一个筛选过程选出一个诚实参与方作为可信第三方来完成协议,并将该框架推广到诚实大多数的四方协议. 在这样的计算框架下,作为可信第三方会拥有所有用户的敏感数据,违背了安全多方计算的初衷. 针对此问题,设计了一个基于(2,4)秘密分享的四方机器学习协议,改进 Swift 框架的诚实参与方筛选过程,以确定出 2 个诚实参与方,并通过他们执行一个半诚实的安全两方计算协议,高效地完成计算任务. 该协议将在线阶段的 25%通信负载转移到了离线阶段,提高了方案在线阶段的效率.

关键词 安全多方计算;隐私保护;机器学习;秘密分享;恶意敌手

中图法分类号 TP391

移动互联网、云计算和大数据等技术的快速发展,催生了众多新的服务模式和应用[1],这些服务和应用一方面为用户提供精准化、个性化的服务,给人们的生活带来了极大便利,另一方面又采集了大量用户的相关信息,而所采集信息中往往含有大量隐私数据,包括个人身份信息(如电话号码、身份证号等)、敏感信息(如金融财务、医疗健康等信息),对这些信息的收集、共享、发布、分析与利用等操作会直接或间接地泄露用户隐私[2],给用户带来极大的威胁和困扰.

隐私保护机器学习(privacy preserving machine learning, PPML)是一个蓬勃发展的研究领域,允许机器学习对用户的私人数据进行计算,同时确保数据的隐私安全[3]. 安全多方计算(secure multi-party computation)协议允许多个参与者通过使用同态加密、秘密共享和不经意传输等密码技术,以一种安全的方式完成数据的聚合与计算,因此成为应用在高效并行分布式机器学习中的关键.近年来,随着数据的爆炸式增加,隐私保护机器学习中完成大量卷积、激活函数等非线性运算,这些计算的实现往往需要复杂的安全多方计算协议,并且需要根据具体的功能来选择不同的安全多方计算技术.总之,基于安全多方计算的隐私保护机器学习方案目前处于初步发展时期,在安全模型、通信效率等还存在没有解决的问题,距离真正应用还有一定的距离.

1 相关工作

PPML 方法最早可追溯至 2000 年,Lindell 等人[4]明确提出了允许两方在不泄露自己隐私的前提下,通过协作对联合数据集进行提取挖掘的方法.

传统的安全多方计算协议为了实现恶意敌手模型下的安全性,需要使用茫然传输(oblivious transfer, OT)、认证秘密分享、零知识证明、承诺等工具,使得安全多方计算协议设计严重依赖这些基础工具,效率较低.近年来,恶意敌手模型下基于 Yao 混乱电路的安全两方计算协议,以及基于秘密分享的安全多方计算协议都提出了高效的实现方案[5].

现有的基于安全多方计算协议的隐私保护机器学习方案有 SecureML,ABY,ABY3,Trident,Swift 等.其中 Mohassel 等人[6]提出的 SecureML 安全机器学习框架,由 2 个不合谋的参与方利用秘密分享执行安全两方计算协议.Demmler 等人提出的 ABY 框架[7]利用 Yao 混乱电路来计算分段函数,该框架能够完成 3 种不同分享方式之间的转换,从而完成计算任务.ABY 和 SecureML 两个方案在设计乘法协议时都利用 Beaver 三元组技术,因此方案的通信成本和协议的效率都并不理想.方案 ABY3[8]在半诚实环境和恶意环境下,能够完成三方之间的算术共享、布尔共享和 Yao 共享,并且对 SecureML[6]中的近似定点数乘法进行改进,使协议在三方参与环境下可以使用.但是 ABY3 方案均是专注于半诚实环境下的机器学习框架,在恶意环境下进行位提取,截断的效率都会不同程度地降低[9-10].

Trident 方案[11]提出了最多可以容忍一方腐败的四方参与的隐私保护机器学习协议,在方案中对参与方要求比较高,通过引入一个额外的诚实参与方来提高协议在线阶段的效率,并且协议在安全性方面仅仅满足中止性,方案利用哈希值进行一致性检测,当出现不一致时,协议就会中止.当恶意参与方提供错误的值时,将导致计算中止,并且协议不会有任何的输出.虽然方案中都提出使用 Abort 的安全构建组件,但当出现争议时,不能保证协议一定有输出.只实现中止安全性的安全多方计算协议不能满足安全外包计算的要求,因为用户可能无法获得输出,导致用户的参与度降低.

2021 年,Koti 等人提出的 Swift 方案[12]中通过引入联合消息传递(joint message passing, JMP)原

语,允许 2 个服务器将共有的消息发送给第 3 个服务器,当发现哈希值不一致时,没有中止协议,而是能够识别出诚实的服务器,将诚实方作为可信第三方(trust third party, TTP)服务器来完成计算.同时文中还将协议扩展到了第 4 个参与方,设计环境依旧是基于诚实大多数,在调用三方筛选协议进行哈希值一致性检测出现争议时,由于最多只有一个恶意方,因此恶意方只可能出现在 3 个服务器中,Swift 方案将未参与筛选协议的第 4 个服务器作为 TTP,参与方就会将消息发送给 TTP,虽然方案能够保证在有恶意参与方的情况下保证用户一定可以获得输出结果,但对于作为 TTP 的参与方来说,就会掌握所有用户的敏感数据信息,不符合安全多方计算协议的初衷.

本文提出了一个基于秘密分享的高效隐私保护四方机器学习方案,在至多存在一个恶意敌手模型中,当协议出现争议时,能够准确从 4 个参与方中确定出 2 个诚实参与方,执行安全两方计算协议.本文设计的隐私保护机器学习方案主要贡献包括 2 个方面:

1) 结合秘密分享和安全多方计算,本文提出了一个在环 $\mathbb{Z}_{2^\ell}$ 上具有主动安全性的四方参与的诚实大多数环境下的高效隐私保护机器学习方案.当协议出现争议时,能够准确识别出诚实的 2 个参与方,从而保证一定有输出结果,避免了由一个参与方掌握所有用户的相关信息,提高了方案的隐私安全要求.

2) 通过对方案的效率分析和对比,本文的四方协议与仅提供半诚实/被动安全性的高效诚实多数三方协议的性能相似,这表明添加第四方是实现主动安全而不损害性能的有效方法.其中乘法协议只需要 3 个参与方在在线阶段处于活跃状态,提高了在线阶段 25%的通信.

2 预备知识

2.1 秘密分享

秘密分享是一个非常重要的基础原语[13-15],是许多安全多方计算协议的重要构造模块.如表 1 所示,4 个参与方 $P=\{P_0,P_1,P_2,P_3\}$ 分享秘密 v 时的 3 种方式,这些分享都是在算数和布尔环上的.为了方便非交互通信,参与方利用 Trident 方案中的功能函数 $\mathcal{F}_{\text{setup}}$ 为每个参与方建立预共享的随机密钥.参与方在分享秘密值 v 时,可以调用共享的密钥设置函数 $\mathcal{F}_{\text{setup}}$ 获得满足条件的相应参数.

$[\cdot]$-sharing:$v\in\mathbb{Z}_{2^\ell}$ 由 3 个参与方 P_1,P_2,P_3 共同分享.参与方 P_1 拥有 $[v]_1$,参与方 P_2 拥有 $[v]_2$,参与方 P_3 拥有 $[v]_3$ 并且 $v=[v]_1+[v]_2+[v]_3$.

$\langle\cdot\rangle$-sharing:$v\in\mathbb{Z}_{2^\ell}$ 由 3 个参与方 P_1,P_2,P_3 共同分享.如果参与方 P_1,P_2,P_3 依次拥有的值为 $(v_2,v_3),(v_1,v_3),(v_1,v_2)$,并且 $v=v_1+v_2+v_3$.

$[\![\cdot]\!]$-sharing:$v\in\mathbb{Z}_{2^\ell}$ 由 4 个参与方 P_0,P_1,P_2,P_3 分享,如果存在 $\lambda_v,m_v\in\mathbb{Z}_{2^\ell}$ 并且满足 $\lambda_v=\lambda_{v,1}+\lambda_{v,2}+\lambda_{v,3}$,$m_v=v+\lambda_v$,参与方 P_1,P_2,P_3 都清楚地知道 m_v.

Table1 Secret Sharing Semantics

表 1 秘密分享形式

参与方	$[v]$	$\langle v\rangle$	$[\![v]\!]$
P_0			$(\lambda_{v,1},\lambda_{v,2},\lambda_{v,3})$
P_1	$[v]_1$	(v_2,v_3)	$(m_v,\lambda_{v,2},\lambda_{v,3})$
P_2	$[v]_2$	(v_1,v_3)	$(\lambda_{v,1},m_v,\lambda_{v,3})$
P_3	$[v]_3$	(v_1,v_2)	$(\lambda_{v,1},\lambda_{v,2},m_v)$

2.2 JMP 原语

Swift 方案中能够实现输出可达性的关键在于引入了 JMP 原语.$JMP_{3\text{PC}}$ 协议在 2021 年由 Koti 等人提出.JMP 原语允许 2 个服务器将公共消息中继到第 3 个服务器,以便中继成功或识别诚实服务器(或冲突对).如图 1 所示为 3 方参与的 JMP 原语的理想功能图.

$\mathcal{F}_{\text{jmp}}$ 与服务器 $\mathcal{P}$ 和敌手 $\mathcal{S}$ 进行交互.

步骤 1.$\mathcal{F}_{\text{jmp}}$ 从 P_s 接收 $(Input,v_s)$,其中 $s\in\{i,j\}$,同时它从 $\mathcal{S}$ 接收 $(Select,ttp)$.这里 ttp 表示 $\mathcal{S}$ 服务器想要成为 TTP.令 $P^*\in\mathcal{P}$ 表示被 $\mathcal{S}$ 破坏的服务器.

步骤 2.如果 $v_i=v_j$ 且 $ttp=\perp$,则设置 $msg_i=msg_j=\perp$,并转到步骤 5.

步骤 3. 如果 $ttp\in\mathcal{P}\backslash\{P^*\}$,则设置 $msg_i=msg_j=msg_k=ttp$.

步骤 4.否则,将 TTP 设置为具有最小索引的诚实服务器.设置 $msg_i=msg_j=msg_k=ttp$.

步骤 5.对于 $s\in\{0,1,2\}$,将 $(Output,msg_s)$ 发送给 P_s.

Fig. 1 Ideal functionality for $JMP_{3\text{PC}}$ primitive

图 1 $JMP_{3\text{PC}}$ 原语的理想功能

如图 2 所示,$JMP_{3\text{PC}}$ 协议内容简要来说,有 3 个服务器参与协议,其中最多只有一个恶意服务器,通过 $JMP_{3\text{PC}}$ 协议可以确保输入一致性,并且当发生争议时,确保协议一定有输出结果.其中 P_i,P_j 作为发送方,P_k 作为接收方.P_i 发送值 v 给 P_k,P_j 发送值 $H(v)$ 给 P_k,接收方 P_k 将接收到的值与哈希值进行比较,如果哈希值不一致,通过 $JMP_{3\text{PC}}$ 协

议可以确认出一个参与方作为 TTP 继续计算,并且不需要再次通信.

输入:P_i,P_j 拥有值 v;
输出:P_k 获得值 v.
协议具体内容:发送阶段:P_i 发送 v 给 P_k.验证阶段:P_j 发送 $H(v)$给 P_k.
服务器 P_s 初始化 $b_s=0,s\in\{i,j,k\}$.
如果 P_i 是沉默的,P_k 广播($accuse$,P_i),那么令 $TTP=P_j$.同理如果 P_j 是沉默的,那么令 $TTP=P_i$.如果 P_k控诉 P_i,P_j,那么 $TTP=P_i$.否则,P_k 接收 $\bar{v}$,如果哈希值一致,设置 $b_k=0$ 否则设置 $b_k=1$,P_k 发送 b_k 给 P_i,P_j,如果 $b_k=0$,协议中止.
如果 P_i 没有从 P_k 接收到 b_k,那么 P_i 广播($accuse$,P_k),并且令 $TTP=P_j$.类似地,如果 P_j 没有从 P_k 接收到 b_k,那么 P_j 广播($accuse$,P_k).如果 P_i,P_j 都广播($accuse$,P_k),那么 $TTP=P_i$;否则,P_i,P_j 均设置 $b_s=b_k$.
P_i,P_j 相互交换它们的值.如果 P_i 没有从 P_j 接收到 b_j,那么就会广播($accuse$,P_j)并令 $TTP=P_k$.对于 P_j 也是类似的.否则,P_i 将 b_i 的值更新为 $b_i\vee b_j$,P_j 将 b_j 的值更新为 $b_j\vee b_i$.
参与方 P_s 广播 $H_s=H(v^*)$,其中 $s\in\{i,j,k\}$.如果 $b_s=1,s\in\{i,j\}$,说明 $v^*=v,v^*=\bar{v}$.否则,如果 P_k 没有广播,协议中止.如果 P_i 或者 P_j 没有广播,那么设置 $TTP=P_k$. 否则,
如果 $H_i\neq H_j$:$TTP=P_k$.
否则如果 $H_i\neq H_k$:$TTP=P_j$.
否则如果 $H_i=H_j=H_k$:$TTP=P_i$.

Fig. 2 JMP_{3PC} protocol
图 2 JMP_{3PC}协议

对于 $s\in\{i,j,k\}$,参与方 P_s 将 $b_s=0$ 作为检测哈希值不一致的标志比特.协议分为发送阶段和验证阶段.当检测到哈希值不一致时,协议将不同的情况进行分类并选择出 TTP.如果 P_i 是沉默的,P_k 没有接收到发送的 v,P_k 广播($accuse$,P_i),那么 P_k 和 P_i 中必定有一个是恶意的参与方,这时选择 $TTP=P_j$.类似地,P_k 广播($accuse$,P_j),那么 P_k 和 P_j 中必定有一个是恶意的参与方,这时选择 $TTP=P_i$.如果当 P_k 收到不一致的消息对(v,$H(v^*)$)时,设定 $b_k=1$,并将 b_k 发送给 P_i,P_j,由这两方通过交换不一致标志比特相互进行交叉检验,若从 P_k 收到的或者从其他发送方接收到的比特为 1,则这两方将自己的不一致标志比特设定为 1.当服务器的不一致标志比特为 1 时,服务器会广播相应值的 Hash 结果.P_k 的值来自它从 P_i 接收到的值.接下来按照具体协议来选出合适的服务器做 TTP.

Swift 方案将 JMP 协议扩展为 4 个参与方 P_i,P_j,P_k,P_l,方案的模型依旧是至多只有一个恶意的参与方.但是当调用三方参与的 JMP_{3PC}协议时,一定可以确定 3 个参与方中存在一个参与方是恶意的,因此将未参与的 JMP_{3PC}协议的第 4 个参与方直接作为 TTP,其他参与方将自己的消息发送给它,由作为 TTP 的参与方完成剩下的所有计算.接下来为了描述方案更加清晰,三方参与的 JMP 协议表示为 JMP_{3PC},四方参与的 JMP 协议表示为 JMP_{4PC}.

2.3 现实-理想范式

令 π 为一个协议,$\mathcal{F}$ 为一个功能函数,令 $\mathcal{C}$ 为攻陷参与方集合,令 Sim 为一个仿真者算法.定义以下 2 个随机变量的概率分布:1)$Real_{\pi}(\kappa,\mathcal{C};x_1,\cdots,x_n)$.在安全参数 κ 下执行协议,其中每个参与方 P_i 都将使用自己的私有输入 x_i 诚实地执行协议.令 V_i 为参与方 P_i 的最终视角,令 y_i 为参与方 P_i 的最终输出.因此可以将输出表示为$\{V_i\mid i\in\mathcal{C}\}$,$(y_1,\cdots,y_n)$.2) $Ideal_{\mathcal{F},Sim}(\kappa,\mathcal{C};x_1,\cdots,x_n)$:计算$(y_1,\cdots,y_n)\leftarrow\mathcal{F}(x_1,\cdots,x_n)$.输出 $Sim(\mathcal{C},\{(x_i,y_i)\mid i\in\mathcal{C}\})$,$(y_1,\cdots,y_n)$.

如果现实世界中攻陷参与方所拥有的视角和理想世界中攻击者所拥有的视角不可区分,那么协议在半诚实攻击者的攻击下是安全的[16-18].

给定协议 π,如果对于任意一个现实世界中的攻击者 $\mathcal{A}$,存在一个满足 $corrupt(\mathcal{A})=corrupt(sim)$ 的仿真者 sim,使得对于诚实参与方的所有输入 $\{x_i\mid i\notin corrupt(\mathcal{A})\}$概率分布 $Real_{\pi,\mathcal{A}}(\kappa;\{x_i\mid i\notin corrupt(\mathcal{A})\})$和 $Ideal_{\mathcal{F},Sim}(\kappa;\{x_i\mid i\notin corrupt(Sim)\})$是在 κ 下不可区分的,则称此协议在恶意攻击者存在的条件下安全地实现了 $\mathcal{F}$.

3 新的基于秘密分享的四方安全多方计算协议

3.1 四方安全多方计算基础协议

3.1.1 秘密分享协议

参与方通过调用秘密分享协议$\Pi_{Sh}(P_i,v)$产生关于$[\![v]\!]$的秘密分享,如图 3 所示.秘密分享协议分为在线阶段和离线阶段.

当执行秘密分享协议$\Pi_{Sh}(P_0,v)$时,在离线阶段:参与方 P_0 持有值 v,使用预先共享的密钥来生成参数 $\lambda_{v,j}$,并且 $\lambda_v=\lambda_{v,1}+\lambda_{v,2}+\lambda_{v,3}$,参与方 P_0 持有份额为$(\lambda_{v,1},\lambda_{v,2},\lambda_{v,3})$,参与方 P_1 持有份额为$(\lambda_{v,2},\lambda_{v,3})$,参与方 P_2 持有份额为$(\lambda_{v,1},\lambda_{v,3})$,参与方 P_3 持有份额为$(\lambda_{v,1},\lambda_{v,2})$.在在线阶段:参与方 P_0 计算 m_v,并发送给 P_1,P_0,参与方 P_1 通过调用 JMP_{3PC}协议发送给 P_2,P_0,参与方 P_1 通过调用 JMP_{3PC}协议发送给 P_3.因此各参与方拥有的份额为参与方 P_0 持有份额为$(\lambda_{v,1},\lambda_{v,2},\lambda_{v,3},m_v)$,参与

方 P_1 持有的份额为($\lambda_{v,2}$,$\lambda_{v,3}$,m_v),参与方 P_2 持有的份额为($\lambda_{v,1}$,$\lambda_{v,3}$,m_v),参与方 P_3 持有的份额为($\lambda_{v,1}$,$\lambda_{v,2}$,m_v).

协议 $\Pi_{\mathrm{Sh}}(P_i,v)$
离线阶段
如果 $P_i=P_0$,参与方 $P\backslash\{P_j\}$共同选取 $\lambda_{v,j}$,其中 $j\in\{1,2,3\}$,P 指的是所有参与方 P_0,P_1,P_2,P_3.
如果 $P_i=P_k$,其中 $k\in\{1,2,3\}$,参与方共同选取 $\lambda_{v,k}$,同时,参与方 $P\backslash\{P_j\}$ 共同选取 $\lambda_{v,j}\in\{1,2,3\}\backslash\{k\}$.
在线阶段
如果 $P_i=P_0$,P_0 计算 $m_v=v+\lambda_v$,并且发送给 P_j.P_i,P_j 调用 JMP 协议发送给 $P_k(i\neq j\neq k)$.
如果 $P_i=P_k$,其中 $k\in\{1,2,3\}$.P_k 计算 $m_v=v+\lambda_v$,并且发送给 P_j.P_j,P_k 调用 JMP_{3PC}协议发送给 P_l.

Fig. 3 $[\![\cdot]\!]$-sharing of a value v by party P_i
图 3 参与方 P_i 分享值 v

当执行秘密分享协议 $\Pi_{\mathrm{Sh}}(P_1,v)$时,在离线阶段:参与方 P_1 拥有值 v,使用预先共享的密钥来生成参数 λ,参与方 P_0 持有份额为($\lambda_{v,1}$,$\lambda_{v,2}$,$\lambda_{v,3}$),参与方 P_1 持有份额为($\lambda_{v,1}$,$\lambda_{v,2}$,$\lambda_{v,3}$),参与方 P_2 持有份额为($\lambda_{v,1}$,$\lambda_{v,3}$),参与方 P_3 持有份额为($\lambda_{v,1}$,$\lambda_{v,2}$).在在线阶段:参与方 P_1 计算 m_v,并且发送给 P_2,P_1,参与方 P_2 通过调用 JMP_{3PC}协议发送给 P_3.因此各参与方拥有的份额为参与方 P_0 持有份额为($\lambda_{v,1}$,$\lambda_{v,2}$,$\lambda_{v,3}$),参与方 P_1 持有份额为($\lambda_{v,1}$,$\lambda_{v,2}$,$\lambda_{v,3}$,m_v),参与方 P_2 持有份额为($\lambda_{v,1}$,$\lambda_{v,3}$,m_v),参与方 P_3 持有份额为($\lambda_{v,1}$,$\lambda_{v,2}$,m_v).参与方 P_2,P_3 执行协议 $\Pi_{\mathrm{Sh}}(P_i,v)$的过程和参与方 P_1 类似,这里不再详细展开了.

如图 4 所示,执行协议 $\Pi_{\mathrm{ash}}(P_0,v)$,参与方 P_0 可以在离线阶段计算值 v 的$\langle\cdot\rangle$-sharing.参与方 P_0 拥有值 v,参与方利用提前约定好的函数共同生成相关的参数.参与方 P_1 拥有份额为(v_2,v_3),参与方 P_2 拥有份额为(v_1,v_3),参与方 P_3 拥有份额为(v_1,v_2).注意在文中第 4 节乘法截断协议中完成 r 不同秘密分享方式的本地转换中可以用到.

协议 $\Pi_{\mathrm{ash}}(P_0,v)$
离线阶段:
参与方 P_2,P_3 共同选取 $v_1\in\mathbb{Z}_{2^\ell}$,$P_1$,$P_3$ 共同选取 $v_2\in\mathbb{Z}_{2^\ell}$.
P_0 根据自己拥有的 v 计算 $v_3=-(v+v_1+v_2)$,将 v_3 发送给 P_1.P_0,P_1 调用 JMP_{3PC}协议将 v_3 发送给 P_2.

Fig. 4 $\langle\cdot\rangle$-sharing of a value v by party P_0
图 4 参与方 P_0 分享值 v

3.1.2 重构协议

如图 5 所示,在重构协议 Π_{Rec}中,每个参与方可以从其他 2 个参与方中依次接收到自己缺失的份额和对应缺失份额的哈希值,参与方通过调用 JMP_{3PC}协议进行一致性输入检测,如果一致的话,那么该参与方可以根据自己持有的份额和收到的自己缺失的份额计算值 v.如果收到的值不一致的话,则由 JMP_{3PC}协议选出的 TTP 和未参与 JMP_{3PC}协议的参与方进行安全两方计算来重构值 v.例如,P_0,P_2 调用 JMP_{3PC}协议发送 $\lambda_{v,1}$给 P_1.如果 JMP_{3PC}协议的执行结果是 $TTP=P_0$,则由 P_0 和 P_3 来计算值 v.

在线阶段:
P_0,P_2 调用 JMP_{3PC}协议发送 $\lambda_{v,1}$给 P_1.
P_0,P_3 调用 JMP_{3PC}协议发送 $\lambda_{v,2}$给 P_2.
P_0,P_1 调用 JMP_{3PC}协议发送 $\lambda_{v,3}$给 P_3.
P_1,P_2 调用 JMP_{3PC}协议发送 m_v 给 P_0.
P_i 计算 $v=m_v-\lambda_{v,1}-\lambda_{v,2}-\lambda_{v,3}$,其中 $i\in\{0,1,2,3\}$.

Fig. 5 Reconstruction protocol
图 5 重构协议

3.1.3 乘法协议

首先简单描述一下加法协议,对于输入 x,y,参与方可以利用秘密分享方案的线性属性本地完成方案中的份额计算,即对于 $z=x+y$ 来说,可以本地计算份额$[z]=[x]+[y]$.接下来描述乘法协议,在协议执行过程中需要参与方进行交互,具体内容如图 6 所示:

输入:$[x]$和$[y]$;
输出:$[xy]$.
离线阶段:
参与方 $P\backslash\{P_j\}$共同选取参数 $\lambda_{z,j}$,其中 $j\in\{1,2,3\}$.详细来说,P_0 拥有($\lambda_{z,1}$,$\lambda_{z,2}$,$\lambda_{z,3}$),P_1 拥有($\lambda_{z,2}$,$\lambda_{z,3}$),P_2 拥有($\lambda_{z,1}$,$\lambda_{z,3}$),P_3 拥有($\lambda_{z,1}$,$\lambda_{z,2}$).
参与方调用协议 Π_{zero},参与方 P_0,P_1,P_2,P_3 获得相关的随机数 A,B,Γ.
P_0,P_1 计算 $\gamma_{xy,2}=\lambda_{x,2}\lambda_{y,2}+\lambda_{x,2}\lambda_{y,3}+\lambda_{x,3}\lambda_{y,2}+A$.
P_0,P_2 计算 $\gamma_{xy,3}=\lambda_{x,3}\lambda_{y,3}+\lambda_{x,3}\lambda_{y,1}+\lambda_{x,1}\lambda_{y,3}+B$.
P_0,P_3 计算 $\gamma_{xy,1}=\lambda_{x,1}\lambda_{y,1}+\lambda_{x,1}\lambda_{y,2}+\lambda_{x,2}\lambda_{y,1}+\Gamma$.
P_2,P_0 调用 JMP_{3PC}协议将 $\gamma_{xy,3}$发送给 P_1.
P_3,P_0 调用 JMP_{3PC}协议将 $\gamma_{xy,1}$发送给 P_2.
P_1,P_0 调用 JMP_{3PC}协议将 $\gamma_{xy,2}$发送给 P_3.
在线阶段:$m'_z=m_z-m_xm_y$.
P_1,P_3 计算 $m'_{z,2}=-\lambda_{x,2}m_y-\lambda_{y,2}m_x+\gamma_{xy,2}+\lambda_{z,2}$.
P_2,P_1 计算 $m'_{z,3}=-\lambda_{x,3}m_y-\lambda_{y,3}m_x+\gamma_{xy,3}+\lambda_{z,3}$.
P_3,P_2 计算 $m'_{z,1}=-\lambda_{x,1}m_y-\lambda_{y,1}m_x+\gamma_{xy,1}+\lambda_{z,1}$.
P_2,P_3 调用 JMP_{3PC}协议将 $m'_{z,1}$发送给 P_1.
P_3,P_1 调用 JMP_{3PC}协议将 $m'_{z,2}$发送给 P_2.
P_1,P_2 调用 JMP_{3PC}协议将 $m'_{z,3}$发送给 P_3.
P_i 计算 $m_z=(m'_{z,1}+m'_{z,2}+m'_{z,3})+m_xm_y=m'_z+m_xm_y$.

Fig. 6 Multiplication protocol
图 6 乘法协议

乘法协议执行分为 2 个阶段:离线阶段和在线阶段.在离线阶段,需要参与方本地计算 γ_{xy} 的分享.其中 $\gamma_{xy}=\lambda_x\lambda_y$.通过调用 JMP_{3PC}协议来验证其他

参与方发送过来的份额的正确性.在计算过程中,通过调用Π_{zero}协议[11]产生随机数目的是随机分配份额,以防止隐私泄露,注意这里的$A+B+\Gamma=0$.在在线阶段,协议的目的主要是计算m_z,其中$m_z=z+\lambda_z=xy+\lambda_z=(m_x-\lambda_x)(m_y-\lambda_y)+\lambda_z=m_xm_y-\lambda_xm_y-\lambda_y\lambda_x+\lambda_x\lambda_y+\lambda_z$.

参与方P_1,P_2,P_3本地计算m'_z的份额.调用$JMP_{3\text{PC}}$协议来验证参与方P_1,P_2,P_3发送过来的m'_z份额是否正确.通过接受到的份额,参与方P_1,P_2,P_3在本地计算加上m_xm_y,就可以获得m_z.

3.2 四方协议框架

本文方案的参与方是由4个服务器$P=\{P_0,P_1,P_2,P_3\}$组成,通过同步网络中私有和真实通道连接,最多容忍一个静态恶意敌手.

本文设计的四方协议如图7所示,当利用哈希值进行一致性检测存在争议时,通过调用$JMP_{3\text{PC}}$三方协议一定可以找出一个$TTP=P_i$,由P_i和未参与$JMP_{3\text{PC}}$协议的服务器进行下一步的计算.4个参与方P_0,P_1,P_2,P_3利用复制秘密共享,对秘密v进行分享,执行秘密分享协议$\Pi_{\text{Sh}}(P_0,v)$后,每个参与方持有的份额是:参与方P_0持有份额为$(\lambda_{v,1},\lambda_{v,2},\lambda_{v,3})$,参与方$P_1$持有份额为$(\lambda_{v,2},\lambda_{v,3},m_v)$,参与方$P_2$持有份额为$(\lambda_{v,1},\lambda_{v,3},m_v)$,参与方$P_3$持有份额为$(\lambda_{v,1},\lambda_{v,2},m_v)$.当执行重构协议,为了确保获得一致性输入,参与方会调用$JMP_{3\text{PC}}$来发送份额,并进行一致性检测,当检测出不一致的情况时,假设调用$JMP_{3\text{PC}}$协议的是P_0,P_1,P_2三个参与方,并且输出结果为$TTP=P_2$时,那么由P_2和P_3来执行安全两方计算协议,并将秘密值v重构出来.其他协议也是同样的过程,这里不再一一展开描述.

预处理阶段:

服务器调用协议$\mathcal{F}_{\text{setup}}$生成公钥[7].参与方$P_s$设置$s=\text{abort}$,其中$s=\{0,1,2,3\}$.当参与方执行$JMP_{3\text{PC}}$协议时,需要先将$s$设置为$s=\text{continue}$,当所有参与方的状态均为在线服务状态时,开始执行协议Π_{sh},Π_{mult},Π_{rec}的离线阶段.

在线阶段:

如果调用$JMP_{3\text{PC}}$协议识别出TTP,由其和未参与协议的参与方继续下一步的计算.否则,服务器执行Π_{sh},Π_{mult},Π_{rec}协议的在线阶段.

输入分享:加法和常数乘法的线性运算在本地就可以完成.

乘法协议:执行协议Π_{mult}的在线阶段.

验证阶段:

通过调用$JMP_{3\text{PC}}$协议来实现,如果在验证过程中,识别出TTP,将结果广播给所有服务器.

重构阶段:

如果验证成功,而且没有TTP标识,则服务器执行协议Π_{rec}的在线阶段.

Fig. 7 Complete 4PC protocol

图7 完整的4PC协议

4 机器学习隐私保护组成模块

为了执行隐私保护机器学习,我们需要有效地实例化3个组件,其中主要包括共享截断、安全比较、非线性激活函数.

4.1 共享截断

协议在ABY3,Trident协议的基础上进行了改进.ABY3方案是在评估乘法门之后对份额进行截断,以较高的概率保留基础值;Trident方案通过不使用任何的布尔电路,从而将离线阶段的复杂性降到了常数.本文在2个协议的基础上,在满足诚实大多数的条件下,结合$JMP_{3\text{PC}}$协议,保证在线阶段参与方可以接受到相应的份额,最后一定可以得到正确的输出结果.

协议的具体内容如图8所示.首先协议执行离线阶段,在执行协议$\Pi_{\text{mult}}(x,y,z)$的离线阶段后,参与方本地计算r并进行截断后获得r^t.参与方执行协议$\Pi_{\text{ash}}(P_0,r^t)$,获得$\langle r^t\rangle$,并通过验证$H(m_1+m_2)\neq H(c)$等式是否成立来判断份额是否正确.协议执行在线阶段,与乘法协议在线阶段类似,z的截断值可以由$[\![z^t]\!]=[\![(z-r)^t]\!]+r^t$获得.

离线阶段:

参与方执行协议$\Pi_{\text{mult}}(x,y,z)$的离线阶段,但不用生成λ_z.参与方本地共同选取如下参数$P_0:(r_1,r_2,r_3)$;$P_1:(r_2,r_3)$;$P_2:(r_1,r_3)$;$P_3:(r_1,r_2)$.P_0本地计算$r=r_1+r_2+r_3$,并将r进行截断获得r^t,执行协议$\Pi_{\text{ash}}(P_0,r^t)$产生关于$r^t$的$\langle\cdot\rangle$-sharing.参与方依次得到的份额是$\langle r^t\rangle_{P_1}=(r_2^t,r_3^t)$,$\langle r^t\rangle_{P_2}=(r_1^t,r_3^t)$,$\langle r^t\rangle_{P_3}=(r_1^t,r_2^t)$.

参与方验证$\langle r^t\rangle$是否正确通过以下步骤:选取一个随机数c,并且计算$m_1=r_2-2^dr_2^t-r_{d,2}+c$.$P_1$发送$(m_1,H(c))$给$P_2$.$P_2$计算$m_2=(r_1+r_2)-2^d(r_1^t+r_3^t)-(r_{d,1}+r_{d,3})$.如果$H(m_1+m_2)\neq H(c)$,就放弃协议的执行.参与方设置$m_{r^t}=0$,$\langle\lambda_{r^t}\rangle=\langle r^t\rangle$,参与方本地可以将$\langle r^t\rangle$转换为$[\![r^t]\!]$.

在线阶段:

参与方本地计算:

P_1,P_3计算$[z']_2=-\lambda_{x,2}m_y-\lambda_{y,2}m_x+\gamma_{xy,2}-r_2$;

P_2,P_1计算$[z']_3=-\lambda_{x,3}m_y-\lambda_{y,3}m_x+\gamma_{xy,3}-r_3$;

P_3,P_2计算$[z']_1=-\lambda_{x,1}m_y-\lambda_{y,1}m_x+\gamma_{xy,1}-r_1$.

P_2,P_3调用$JMP_{3\text{PC}}$协议发送$[z']_1$给P_1;

P_1,P_3调用$JMP_{3\text{PC}}$协议发送$[z']_2$给P_2;

P_2,P_1调用$JMP_{3\text{PC}}$协议发送$[z']_3$给P_3.

P_i计算$(z-r)=[z']_1+[z']_2+[z']_3+m_xm_y$;

P_1,P_2,P_3本地截断$(z-r)$获得$(z-r)^t$,执行秘密分享协议产生$[\![(z-r)^t]\!]$.

参与方本地计算$[\![z^t]\!]=[\![(z-r)^t]\!]+r^t$.

Fig. 8 Protocol $\Pi_{\text{MultTr}(x,y,z)}$

图8 协议$\Pi_{\text{MultTr}(x,y,z)}$

4.2 安全比较

参与方检测 $x<y$ 是否成立.在方案中常用到的简单方法就是输入 2 个算术分享 x,y,计算 $x-y$.通过判断符号位来判断大小.协议的操作步骤关键在于调用位提取(bit extraction)协议.从给定的算术份额中提取出一位,并输出对应该位的布尔份额.P_0,P_3 产生布尔分享 $y=\lambda_{v,1}+\lambda_{v,2}$.$P_1,P_2$ 产生布尔分享 $x=m_v-\lambda_{v,3}$.各方利用文献 ABY3 中类似方法获得 MSB.这里不再展开描述.

4.3 激活函数

在隐私保护机器学习中最常用的 2 个激活函数为 rectified linear Unit(ReLU)和 Sigmoid(sig).

1) ReLU

ReLU 函数可以定义为 $relu(v)=\max(0,v)$.展开来说,$relu(v)=\bar{b}\cdot v$,如果 $v<0$,则位 $b=1$,否则 $b=0$.为了产生$[\![relu(v)]\!]$,参与方执行位提取协议获得$[\![b]\!]^B$,$\bar{b}$ 代表 b 的补码,设置 $\beta_{\bar{b}}=1\oplus\beta_b$,$\bar{b}$ 的 $[\![\cdot]\!]^B$-sharing 可以本地计算获得.

2) Sigmoid

在协议中,我们主要从函数定义可以看出,函数 Sigmoid 类似于 Trident 方案.其中 $sig(v)=\bar{b}_1b_2\times\left(v+\frac{1}{2}\right)+\bar{b}_2$,如果 $v+1/2<0,b_1=1$,如果 $v-1/2<0,b_2=1$.为了计算$[\![sig(v)]\!]$,服务器以与 ReLU 中类似的方式进行,因此细节方面我们这里就不展开描述了.

$$sig(v)=\begin{cases}0, & v<-\frac{1}{2},\\ v+\frac{1}{2}, & -\frac{1}{2}\leqslant v\leqslant\frac{1}{2},\\ 1, & v>\frac{1}{2}.\end{cases}$$

5 安全分析

5.1 正确性

在截断协议Π_{MultTr}的离线阶段,参与方 P_0 产生$[\![r^t]\!]$分享,其他参与方可以通过验证等式$H(m_1+m_2)=H(c)$是否成立,决定协议是否执行.通过协议中的描述,我们已知 $m_1=r_2-2^dr_2^t-r_{d,2}$,$m_2=(r_1+r_3)-2^d(r_1^t+r_3^t)-(r_{d,1}+r_{d,3})$.其中,$r=2^dr^t+r_d$,$r^t$ 表示 r 的截断值,r_d 表示 r 后 d 位.

$$\begin{aligned}m_1+m_2&=(r_1+r_2+r_3)+2^d(r_1^t+r_2^t+r_3^t)-\\&(r_{d,1}+r_{d,2}+r_{d,3})+c=(r)-(2^dr^t+r_d)+\\&c=0+c=c.\end{aligned}$$

5.2 协议安全证明

本文提供的安全性是基于理想世界或现实世界模拟来展开的描述[19-20].现实世界中参与方至多有一个是恶意服务器,我们用 $\mathcal{A}$ 来表示现实世界的敌手,用 $\mathcal{S}$ 来表示理想世界的敌手.证明来自于敌手的模拟视图和现实视图是不可区分的.

首先本节提供 $JMP_{3\text{PC}}$ 原语的安全证明,该协议是实现输出可达性的关键构造.在协议中 $\mathcal{F}_{\text{jmp}}$ 表示理想功能,让 $\mathcal{S}_{\text{jmp}}^{P_i}$ 表示被腐化的 $P_s\in\mathcal{P}$ 相关的模拟器.其中被腐化的参与方我们分别从发送者和接收者 2 种情况讨论,如图 9 和 10 所示:

$\mathcal{S}_{\text{jmp}}^{P_i}$ 初始化 $ttp=\perp$,代表 P_k 从敌手 $\mathcal{A}$ 接收 v_i.

敌手没有成功发送值 v_i,$\mathcal{S}_{\text{jmp}}^{P_i}$ 广播$(accuse,P_i)$,设置 $ttp=P_j$,$v_i=\perp$,并直接跳到最后一步.

否则,它检查 $v_i=v$ 是否成立,其中 v 是基于与敌手 $\mathcal{A}$ 交互通过执行 $\mathcal{S}_{\text{jmp}}^{P_i}$ 协议并使用共享密钥的知识计算的值.如果值相等,则 $\mathcal{S}_{\text{jmp}}^{P_i}$ 设置 $b_k=0$,否则设置 $b_k=1$,并代表 P_k 将其发送给 $\mathcal{A}$.

如果敌手广播$(accuse,P_k)$,设置 $ttp=P_j$,$v_i=\perp$,并直接跳到最后一步.

$\mathcal{S}_{\text{jmp}}^{P_i}$ 代表 P_j 计算 b_j 并将其发送给 $\mathcal{A}$,并代表诚实的 P_j 从 $\mathcal{A}$ 接收 $b_{\mathcal{A}}$.

如果 $\mathcal{S}_{\text{jmp}}^{P_i}$ 没有收到代表 P_j 的 $b_{\mathcal{A}}$,则广播$(accuse,P_k)$,设置 $v_i=\perp$,$ttp=P_k$.如果 $\mathcal{A}$ 广播$(accuse,P_k)$,设置 $v_i=\perp$,$ttp=P_k$.如果设置了 ttp,则跳到最后一步.

如果 $v_i=v$ 并且 $b_{\mathcal{A}}=1$,$\mathcal{S}_{\text{jmp}}^{P_i}$代表 P_j 广播 $H_j=H(v)$.

如果 $v_i\neq v_j$,$\mathcal{S}_{\text{jmp}}^{P_i}$代表 P_j 和 P_k 依次广播 $H_j=H(v)$和 $H_k=H(v_i)$.如果没有广播,$\mathcal{S}_{\text{jmp}}^{P_i}$设置 $ttp=P_k$.

否则,$\mathcal{A}$ 广播值 $H_{\mathcal{A}}$:如果 $H_{\mathcal{A}}\neq H_j$,$\mathcal{S}_{\text{jmp}}^{P_i}$ 设置 $ttp=P_k$;否则如果 $H_{\mathcal{A}}\neq H_k$,$\mathcal{S}_{\text{jmp}}^{P_i}$ 设置 $ttp=P_j$.

$\mathcal{S}_{\text{jmp}}^{P_i}$ 代表 $\mathcal{A}$,调用 $\mathcal{F}_{\text{jmp}}$ 协议接收$(Input,v_i)$和$(Select,ttp)$.

Fig. 9 Simulator $\mathcal{S}_{\text{jmp}}^{P_i}$ for the case of corrupt P_i

图 9 P_i 腐化情况下模拟器 $\mathcal{S}_{\text{jmp}}^{P_i}$

$\mathcal{S}_{\text{jmp}}^{P_k}$ 初始化 $ttp=\perp$,诚实地计算 v 并分别代表 P_i 和 P_j 将 v 和 $H(v)$ 发送给 $\mathcal{A}$.

如果 $\mathcal{A}$ 广播$(accuse,P_i)$,设置 $ttp=P_j$.否则如果 $\mathcal{A}$ 广播$(accuse,P_j)$,设置 $ttp=P_k$.如果 2 个消息都被广播,设置 $ttp=P_i$.如果设置了 ttp,则跳到最后一步.

$\mathcal{S}_{\text{jmp}}^{P_k}$ 代表 P_i,P_j 从 $\mathcal{A}$ 接收 $b_{\mathcal{A}}$.b_i 代表 P_i 从敌手 $\mathcal{A}$ 接收到的比特,b_j 代表 P_j 从敌手 $\mathcal{A}$ 接收到的比特.

如果 $\mathcal{A}$ 发送比特 $b_{\mathcal{A}}$ 给 P_i 失败,$\mathcal{S}_{\text{jmp}}^{P_k}$广播$(accuse,P_k)$,设置 $ttp=P_j$.P_j 也是相同的操作.如果 P_i,P_j 都广播$(accuse,P_k)$,则设置 $ttp=P_i$.如果设置了 ttp,则跳到最后一步.

如果 $b_i\vee b_j=1$,$\mathcal{S}_{\text{jmp}}^{P_k}$分别代表 P_i,P_j 广播 H_i,H_j,其中 $H_i=H_j=H(v)$.

如果 $\mathcal{A}$ 不广播 $\mathcal{S}_{\text{jmp}}^{P_k}$,设置 $ttp=\perp$.如果 $\mathcal{A}$ 广播一个值 $H_{\mathcal{A}}$:如果 $H_{\mathcal{A}}\neq H_i$,$\mathcal{S}_{\text{jmp}}^{P_k}$ 设置 $ttp=P_j$;否则如果 $H_{\mathcal{A}}=H_i=H_j$,$S_{\text{jmp}}^{P_k}$设置 $ttp=P_i$.

$\mathcal{S}_{\text{jmp}}^{P_i}$ 代表 $\mathcal{A}$,调用 $\mathcal{F}_{\text{jmp}}$ 协议接收$(Input,v_i)$和$(Select,ttp)$.

Fig. 10 Simulator $\mathcal{S}_{\text{jmp}}^{P_k}$ for the case of corrupt P_k

图 10 P_k 腐化情况下模拟器 $\mathcal{S}_{\text{jmp}}^{P_k}$

协议的证明过程简要来说,从输入共享阶段开始,$\mathcal{S}$ 将诚实方的输入设置为 0.模拟器可以从共享协议中提取 $\mathcal{A}$ 的输入.$\mathcal{S}$ 可以获得整个协议的所有输入,因此 $\mathcal{S}$ 可以计算电路的所有中间值和输出.参与方 P_0,P_1,P_2,P_3 中至多有一个恶意参与方.除了图 11 所示协议的安全性证明,其他基本协议的证明也很容易推导出来,这里就不全部展开描述了.

离线阶段:

$\mathcal{S}_{\mathrm{Sh}}^{P_0}$ 执行 $\mathcal{F}_{\mathrm{setup}}$ 协议,并将获得的密钥(k_{01},k_{12},k_{13},k_{012},k_{013},k_{123})和 $k_{\mathcal{P}}$ 提供给 $\mathcal{A}$,通过模拟 $\mathcal{F}_{\mathrm{setup}}$,它产生与 v 相关的 λ 的值.

如果 $P_s=P_0$,$\mathcal{S}_{\mathrm{Sh}}$ 代表 P_j 使用分享的密钥计算 $\lambda_{v,j}$,其中 $j\in\{1,2,3\}$.

如果 $P_s=P_k$,$\mathcal{S}_{\mathrm{Sh}}$ 使用密钥 $k_{\mathcal{P}}$ 计算 $\lambda_{v,k}$.与诚实方相对应的 λ_v 的份额由 $\mathcal{S}_{\mathrm{Sh}}$ 诚实地进行计算.

在线阶段:

如果 $P_s=P_0$,$\mathcal{S}_{\mathrm{Sh}}^{P_0}$ 代表 P_1,P_2,P_3 接收 m_v 并相应地设置 $msg=v$.P_0 作为发送者,执行 $\mathcal{S}_{\mathrm{jmp}}^{P_i}$ 协议.

如果 $P_s\neq P_0$,$\mathcal{S}_{\mathrm{Sh}}^{P_0}$ 代表 P_2 接收 m_v 并相应地设置 $v=0$.P_0 作为接收者,执行 $\mathcal{S}_{\mathrm{jmp}}^{P_k}$ 协议.

Fig. 11 Simulator $\mathcal{S}_{\mathrm{Sh}}$ for the case of corrupt P_0

图 11 P_0 腐化情况下的模拟器 $\mathcal{S}_{\mathrm{Sh}}$

5.3 隐私保护鲁棒性

本文提出隐私保护鲁棒性的 4PC 隐私保护机器学习方案,是针对 JMP 四方协议的扩展,可以更好地保护诚实方的隐私.在协议的执行过程中,正在参与 JMP 协议的 3 个服务器,当接收方接收到的哈希值不一致时,可以确认出恶意方在哪一对服务器中,同时确认出一定是诚实方的参与方作为 TTP,由 TTP 和未参与 JMP 协议的第 4 个服务器进行下一步的计算.因此,该协议可以在恶意参与方破坏的情况下仍然保持正确的输出,同时诚实方不能恢复私有输入.在协议执行的过程中,我们可以准确地判断协议的问题出现在哪一步,能够更好地对服务器的性能进行综合评估,并且当协议中确认 TTP 后会及时广播.

在通过 $JMP_{3\mathrm{PC}}$ 协议确认 TTP 后,TTP 与未参与协议的参与方执行安全两方计算协议.方案能够更好地保护用户隐私,不会将全部的信息以明文的形式由一个参与方保存,同时锁定了可能的恶意方,并且恶意方在整个过程中不会得到额外的消息.协议的调用过程中不会泄露正在计算的私有信息,但仍然允许计算完成.诚实方不存储非协议的消息,在恶意方存在的情况下,能够保持计算的正确性.隐私保护鲁棒性协议的安全性和实用性具有非常重要的意义.

6 方案效率分析

6.1 通信开销

如何降低协议的通信量和轮数复杂度一直都是设计一个更加高效的隐私保护机器学习方案的关键.接下来主要包括本文涉及到的基本协议具体效率分析,如表 2 所示,将本文方案的离线阶段和在线阶段的交互轮数以及通信量与 ABY3 方案进行了详细的对比.通过对比可以发现,本文提出的改进之后的四方隐私保护方案降低了通信量.

Table 2 Communication Overhead of Protocol

表 2 协议的通信开销

协议	方案	离线阶段		在线阶段	
		通信量/b	轮数	通信量/b	轮数
Π_{Mult}	ABY3	4ℓ	1	11ℓ	1
	本文	3ℓ	1	3ℓ	1
Π_{MultTr}	ABY3	$96\ell-42d-84$	$2\ell-2$	12ℓ	1
	本文	6ℓ	2	3ℓ	1
Π_{ReLU}	ABY3	60ℓ	3	45ℓ	$3+\log\ell$
	本文	$8\ell+2$	3	$8\ell+2$	4
Π_{Sigmoid}	ABY3	$108\ell+12$	3	$81\ell+9$	$4+\log\ell$
	本文	$15\ell+7$	3	$16\ell+7$	5

注:"ℓ"代表在环中的大小(以位为单位).

从 2 个方面将本文提出的方案进行分析,包括线性计算协议和非线性计算协议.1)关于秘密分享的加法协议和常数的乘法是可以本地完成份额的计算,因此其线性属性可以以非交互方式来执行协议.2)秘密分享协议 Π_{Sh} 在离线阶段通过使用 $\mathcal{F}_{\mathrm{setup}}$ 生成共享密钥.在在线阶段,当 $P_i=P_0$ 时,协议 Π_{ash} 计算并发送值 m_v 给其他参与方,并且调用 JMP 协议,确认收到了正确的 m_v,因此需要 2 轮交互和通信量是 2ℓ.协议 Π_{ash} 可以完整地在离线阶段执行.P_0 计算 v_3 发送给 P_1.P_0,P_1 调用 JMP 协议发送给 P_2.因此需要一轮交互和通信量是 2ℓ 位.对于发送给其他参与方也是类似的情况.3)重构协议 Π_{Rec} 在在线阶段每一个参与方接收自己的缺失的份额并进行验证,因此需要的一轮交互和通信量为 4ℓ 位.4)乘法协议的执行分为 2 个阶段,在离线阶段,P_1,P_2,P_3 需要与其他参与方交互获得 γ_{xy} 的份额,同理,离线阶段,P_1,P_2,P_3 需要获得 m'_z 的份额,因此需要的一轮交互和通信量是 3ℓ 位.

机器学习的组成模块中进行非线性计算时,

常用到截断协议、安全比较协议、激活函数等.本文涉及到的基本协议包括:1)协议Π_{MultTr}在离线阶段,需要调用Π_{ash}协议来产生关于γ^t的分享,通过判断哈希值是否相等来验证分享是否正确,因此需要2轮交互和6ℓ位的通信量.2)协议Π_{ReLU}在离线阶段需要3轮交互和$8\ell+2$位的通信量,在在线阶段需要4轮交互和$8\ell+2$位的通信量.3)协议Π_{Sigmoid}在离线阶段需要3轮交互和$15\ell+7$位的通信量,在在线阶段需要5轮交互和$16\ell+7$位的通信量.

6.2 方案对比

结合安全多方计算工具来实现隐私保护机器学习,不同的安全多方计算协议可以适用于不同的场景,因此出现了许多使用不同安全多方计算原语组合构建机器学习隐私的保护方案.如表3所示:

Table 3 Comparison of Privacy Preserving Machine Learning Schemes Based on Multi-party Computation

表3 多方计算的隐私保护机器学习相关方案的比较

方案	参与方个数	模型	特征
ABY	2	半诚实	提供了可以实现不同分享之间转换的ABY架构
SecureML	2	半诚实	改进近似定点乘法协议
ABY3	3	半诚实、恶意	提出了在三方参与的半诚实环境和恶意环境的新方案,需要使用三元组来完成乘法协议
Trident	4	恶意	额外添加一个诚实方来提高在线阶段的效率,在乘法协议中不需要使用三元组
Swift	3,4	恶意	通过筛选协议选出一个诚实方作为可信第三方,其他参与方将数据发送该参与方,由该参与方完成计算
本文	4	恶意	不需要额外引进诚实方,通过筛选协议确定出2个诚实的参与方,执行安全的两方计算协议,更好地保护用户的数据隐私

为了更好地理解本文提出方案的实现功能和应用范围,本文提供了多方参与的隐私保护机器学习相关方案在参与方个数、模型及方案的主要特征进行了对比.通过方案对比可以得出:1)ABY和SecureML只满足半诚实环境下的安全多方计算协议;2)Trident方案需要额外引入一个诚实的参与方;3)Swift方案虽然通过筛选协议选出了一个TTP,但是不论是三方还是四方的安全多方计算协议都是其他参与方把所有信息发给TTP继续计算,不符合安全多方计算协议设计的初衷;4)本文提出的四方隐私保护机器学习方案能够在协议发生争议时,筛选出2个诚实参与方继续执行协议,保证用户可以得到输出结果,不用担心因为恶意敌手的行为而拒绝服务.确定出的2个诚实参与方能够高效地完成安全两方计算协议.

7 总结与展望

近年来,隐私保护机器学习方案在实用性和模型准确性等方面取得了很大的进步,但是仍然有许多问题需要解决.基于安全多方计算的隐私保护机器学习中的模型设计和通信开销一直是研究的重点.本文提出了一个完整的四方参与的隐私保护机器学习方案,遵循诚实大多数的原则,不需要单独引入额外的诚实方,同时能够保证在有恶意参与方的情况下,协议依然能够正确计算保证输出.

下一步的工作主要由3个方面展开:1)关于方案参与方的数量问题,如何将协议的参与方个数扩展到N,并且协议实现的环境不再只是针对至多一个恶意参与方.2)如何在提升安全属性的前提下,将性能开销进一步优化.3)针对不同的应用需求和不同的应用架构,如何更好地保护模型参数信息、降低隐私泄露的风险,都是未来我们需要关注和解决的问题.

作者贡献声明:阎允雪提出了方案的具体思路和实验方案并撰写论文;马铭负责完成实验的对比分析并撰写论文;蒋瀚提出指导意见并修改论文.

参 考 文 献

[1] Liu Junxu, Meng Xiaofeng. Survey on privacy-preserving machinelearning [J]. Journal of Computer Research and Development, 2020, 57(2): 346-362 (in Chinese)
(刘俊旭,孟小峰.机器学习的隐私保护研究综述[J].计算机研究与发展,2020,57(2):346-362)

[2] Song Lei, Ma Chunguang, Duan Guanghan. Machine learning security and privacy: A survey [J]. Chinese Journal of Network and Information Security, 2018, 4(8): 5-15 (in Chinese)
(宋蕾,马春光,段广晗.机器学习安全及隐私保护研究进展[J].网络与信息安全学报,2018,4(8):5-15)

[3] Li Pan, Zhao Wentao, Liu Qiang, et al. Security issues and their counter measuring techniques of machine learning: A survey [J]. Journal of Frontiers of Computer Science and Technology, 2018, 12(2): 171-184 (in Chinese)

(李盼, 赵文涛, 刘强, 等. 机器学习安全性问题及其防御技术研究综述[J]. 计算机科学与探索, 2018, 12(2): 171-184)

[4] Lindell Y, Pinkas B. Privacy preserving data mining [C] // Proc of Annual Int Cryptology Conf. Berlin: Springer, 2000: 36-54

[5] Jiang Han, Xu Qiuliang. Advances in key techniques of practical secure multi-party computation [J]. Journal of Computer Research and Development, 2015, 52(10): 2247-2257 (in Chinese)
(蒋瀚, 徐秋亮. 实用安全多方计算协议关键技术研究进展[J]. 计算机研究与发展, 2015, 52(10): 2247-2257)

[6] Mohassel P, SecMohassel P, Zhang Yupeng. SecureML: A system for scalable privacy-preserving machine learning [C] //Proc of 2017 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2017: 19-38

[7] Demmler D, Schneider T, Zohner M. ABY—A framework for efficient mixed-protocol secure two-party computation [C] //Proc of Network and Distributed System Security Symp. Reston, VA: Internet Society, 2015: 1-15. DOI: 10.14722/ndss.2015.23113

[8] Mohassel P, Rindal P. ABY3: A mixed protocol framework for machine learning [C]//Proc of the 2018 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2018: 35-52

[9] Chaudhari H, Choudhury A, Patra A, et al. ASTRA: High throughput 3PC over rings with application to secure prediction [C] //Proc of the 2019 ACM SIGSAC Conf on Cloud Computing Security Workshop. New York: ACM, 2019: 81-92

[10] Patra A, Suresh A. BLAZE: Blazing fast privacy—Preserving machine learning [J/OL]. arXiv preprint, arXiv: 2005.09042, 2020 [2020-07-20]. https://arxiv.org/abs/2005.09042

[11] Chaudhari H, Rachuri R, Suresh A. Trident: Efficient 4PC framework for privacy preserving machine learning [C] // Proc of the 27th Annual Network and Distributed System Security Symp (NDSS 2020). San Diego, CA: ISOC, 2020: 23-26

[12] Koti N, Pancholi M, Patra A, et al. Swift: Super-fast and robust privacy-preserving machine learning [C/OL] // Cryptology ePrint Archive, Report2020/592, 2020 [2022-03-15]. https://eprint.iacr.org/2020/592

[13] Byali M, Chaudhari H, Patra A, et al. FLASH: Fast and robust framework for privacy preserving machine learning [J]. Proceeding on Privacy Enhancing Technologies, 2020, 2020(2): 459-480

[14] Wagh S, Gupta D, Chandran N. SecureNN: 3-party secure computation for neural network training [J]. Proceedings on Privacy Enhancing Technologies, 2019, 2019(3): 26-49

[15] Keller M, Pastro V, Rotaru D. Overdrive: Making SPDZ great again [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2018: 158-189

[16] Hesamifard E, Takabi H, Ghasemi M, et al. Privacy-preserving machine learning in cloud [C] //Proc of the 2017 on Cloud Computing Security Workshop. New York: ACM, 2017: 39-43

[17] Wagh S, Tople S, Benhamouda F, et al. FALCON: Honest-majority maliciously secure framework for private deep learning [J/OL]. Arxiv preprint, 2020 [2020-07-20]. https://arxiv.org/abs/2004.02229

[18] Lindell Y, Pinkas B. An efficient protocol for secure two-party computation in the presence of malicious adversaries [G] //LNCS 4515: Advances in Cryptology (EUROCRYPT 2007). Berlin: Springer, 2007: 52-78

[19] Bonawitz K, Ivanov V, Kreuter B, et al. Practical secure aggregation for privacy-preserving machine learning [C] // Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 1175-1191

[20] Asharov G, Lindell Y, Schneider T, et al. More efficient oblivious transfer extensions with security for malicious adversaries [G] //LNCS 9056: Advances in Cryptology (EUROCRYPT 2015). Berlin: Springer, 2015: 673-701

Yan Yunxue, born in 1994. PhD candidate. Her main research interests include cryptography and secure multi-party computation.
阎允雪,1994 年生.博士研究生.主要研究方向为密码学、安全多方计算.

Ma Ming, born in 1998. Master candidate. His main research interests include machine learning and secure multi-party computation.
马　铭,1998 年生.硕士研究生.主要研究方向为机器学习和安全多方计算.

Jiang Han, born in 1974. PhD, associate professor, master supervisor. His main research interests include cryptography, information security, especially on secure multi-party computation.
蒋　瀚,1974 年生.博士,副教授,硕士生导师.主要研究方向为密码学、信息安全,特别是安全多方计算.