计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2021.20210626
58(10):2163-2186,2021

安全多方计算及其在机器学习中的应用

郭娟娟[1,2] 王琼霄[1,2] 许 新[1,2] 王天雨[3] 林璟锵[4]
[1](信息安全国家重点实验室(中国科学院信息工程研究所) 北京 100195)
[2](中国科学院大学网络空间安全学院 北京 100049)
[3](华控清交信息科技(北京)有限公司 北京 100084)
[4](中国科学技术大学网络空间安全学院 合肥 230026)
(guojuanjuan@iie.ac.cn)

Secure Multiparty Computation and Application in Machine Learning

Guo Juanjuan[1,2], Wang Qiongxiao[1,2], Xu Xin[1,2], Wang Tianyu[3], and Lin Jingqiang[4]
[1](*State Key Laboratory of Information Security* (*Institute of Information Engineering*, *Chinese Academy of Sciences*), *Beijing* 100195)
[2](*School of Cyber Security*, *University of Chinese Academy of Sciences*, *Beijing* 100049)
[3](*Huakong TsingJiao Information Science* (*Beijing*) *Limited*, *Beijing* 100084)
[4](*School of Cyber Security*, *University of Science and Technology of China*, *Hefei* 230026)

Abstract With the emergence and development of artificial intelligence and big data, large-scale data collection and analysis applications have been widely deployed, which introduces the concern of privacy leakage. This privacy concern further prevents data exchanges among originations and results in "data silos". Secure multiparty computation (MPC) allows multiple originations to perform privacy-preserving collaborative data analytics, without leaking any plaintext data during the interactions, making the data "usable but not visible". MPC technologies have been extensively studied in the academic and engineering fields, and derive various technical branches. Privacy-preserving machine learning (PPML) is becoming a typical and widely deployed application of MPC. And various PPML schemes have been proposed to perform privacy-preserving training and inference without leaking model parameters nor sensitive data. In this paper, we systematically analyze various MPC schemes and their applications in PPML. Firstly, we list various security models and objectives, and the development of MPC primitives (i.e., garble circuit, oblivious transfer, secret sharing and homomorphic encryption). Then, we summarize the strengths and weaknesses of these primitives, and list the corresponding appropriate usage scenarios, which is followed by the thorough analysis of their applications in PPML. Finally, we point out the further research direction on MPC and their applications in PPML.

Key words secure multiparty computation (MPC); garbled circuit; oblivious transfer; secret sharing; homomorphic encryption; privacy-preserving machine learning

摘 要 随着人工智能、大数据等技术的发展,数据采集、数据分析等应用日渐普及,隐私泄露问题越来越严重.数据保护技术的缺乏限制了企业之间数据的互通,导致形成"数据孤岛".安全多方计算(secure

收稿日期:2021-06-11;**修回日期**:2021-07-29
基金项目:国家自然科学基金面上项目(61772518)
This work was supported by the General Program of the National Natural Science Foundation of China (61772518).
通信作者:王琼霄(wangqiongxiao@iie.ac.cn)

multiparty computation, MPC)技术能够在不泄露明文的情况下实现多方参与的数据协同计算,实现安全的数据流通,达到数据"可用不可见".隐私保护机器学习是当前 MPC 技术最典型也是最受关注的应用与研究领域,MPC 技术的应用可以保证在不泄露用户数据隐私和服务商模型参数隐私的情况下进行训练和推理.针对 MPC 及其在隐私保护机器学习领域的应用进行全面的分析与总结,首先介绍了 MPC 的安全模型和安全目标;梳理 MPC 基础技术的发展脉络,包括混淆电路、不经意传输、秘密分享和同态加密;并对 MPC 基础技术的优缺点进行分析,提出不同技术方案的适用场景;进一步对基于 MPC 技术实现的隐私保护机器学习方案进行了介绍与分析;最后进行总结和展望.

关键词 安全多方计算;混淆电路;不经意传输;秘密分享;同态加密;隐私保护机器学习

中图法分类号 TP309

安全多方计算(secure multiparty computation, MPC)起源于姚期智在 1982 年提出的百万富翁问题.在 MPC 中,参与方将各自的秘密数据输入到一个约定函数进行协同计算,即使在一方甚至多方被攻击的情况下,MPC 仍能保证参与方的原始秘密数据不被泄露,并且保证函数计算结果的正确性.自 MPC 理论创立以来,已经衍生出多个技术分支,包括混淆电路、秘密分享、同态加密和不经意传输等.

混淆电路(garbled circuit, GC)是姚期智于 1986 年提出的安全两方计算协议[1],参与方在不知晓他人数据的前提下,使用私有数据共同计算一个用逻辑电路表示的函数.大多数混淆电路方案支持 2 个参与方计算,其性能优化集中于优化单个电路门的密文数量、传输的电路数量等.近年来多参与方计算的混淆电路方案相继提出.混淆电路的优点是能在恒定轮数内完成计算,只涉及开销较小的对称加密运算;缺点是通信量和电路大小呈线性关系,因此不适合计算复杂运算,只适用于比较大小等简单的逻辑运算.

秘密分享(secret sharing, SS)由 Shamir[2] 和 Blakley[3]于 1979 年分别提出,数据拥有方计算秘密数据的份额并将其分发给计算方,计算方对不同秘密的份额计算,得到计算结果的份额.由于秘密拆分方式不同,秘密分享分为基于多项式插值的秘密分享和加性秘密分享(additive secret sharing).加性算术秘密分享能够计算线性运算、加性布尔秘密分享能够计算比较大小等非线性运算.加性秘密分享只需要进行简单的运算,计算开销小,在 MPC 中得到了广泛的应用.秘密分享的缺点是交互轮数和电路深度有关.

不经意传输(oblivious transfer, OT)由 Rabin 于 1981 年提出[4],消息发送方拥有多个消息,接收方获得其中某个值,发送方不知道接收方的选择信息.其衍生技术相关不经意传输(correlated oblivious transfer, COT)可以生成具有关系的随机数.OT, COT 是重要的密码学组件,能够为 GC 传输导线标签、为 SS 生成相关随机数,只使用 OT 技术也能实现 MPC 协议.在 OT 的性能优化历程中,OT 扩展(OT Extension)技术引入对称加密来降低计算开销,静默 OT 扩展(silent OT Extension)技术在本地扩展随机数种子来降低通信开销.

同态加密(homomorphic encryption, HE)的概念由 Rivest 等人[5]于 1978 年提出,可以在无需解密的情况下直接对加密数据执行计算.在发展过程中先后有部分同态加密方案与浅同态加密方案提出,直到 2009 年 Gentry[6]构造出首个全同态加密方案.同态加密的优点是能以最小的通信成本设计轮数较优的 MPC 协议,缺点是乘法同态运算会带来较大的计算和存储开销,目前加法同态在实际中应用较多.

以上 4 类 MPC 技术在计算性能、通信效率和存储开销等方面都具有各自的优势和劣势,单一的 MPC 技术往往不能满足实际应用中计算复杂函数的需求,需要将多种 MPC 技术相结合才能获得性能均衡的实现方案.

目前 MPC 技术最典型的应用场景是隐私保护机器学习(privacy-preserving machine learning, PPML). MPC 在模型训练和推理中可以保护用户数据和模型参数的隐私.近年来,Microsoft, Amazon 和 Google 等公司提供"机器学习即服务"(machine learning as a service, MLaaS),即公司提供训练好的模型来对用户数据计算推断结果.模型数据是公司的重要资产,一旦泄露会带来重大的经济损失,同时用户也不愿意泄露私有数据,MPC 能够在保护双方隐私的前提下完成模型推理.此外,在隐私保护法律法规 GDPR, EFF 和 HIPPA 等的约束下,不同机构间无法交换明文数据,MPC 能够在保证数据集隐私的前提下来完成模型的训练.

PPML 方案涉及多种类型的运算,考虑到参与方的数据规模不同、计算和网络能力不同、安全模型不同,可以在具体场景下组合多种 MPC 技术来实现 PPML 方案.

1 安全模型及目标

根据系统对敌手的容忍程度,MPC 的安全模型可以分为半诚实安全模型(semi-honest model)和恶意安全模型(malicious model).

半诚实安全模型:敌手会按照协议规定进行运算,但试图通过协议中得到的信息挖掘其他参与方的隐私数据,该模型保证用户隐私信息不被敌手获得.

恶意安全模型:敌手不会遵守约定执行协议,会以错误的协议执行结果推断信息,恶意攻击行为也可以是任意的不可推理的.该模型中用户隐私数据不会被敌手获得,且不会因为敌手的恶意行为导致协议错误运行.在恶意安全模型中,要确保准确检测到敌手恶意行为会带来很大开销,因此提出了安全性和开销更均衡的隐蔽安全(covert)模型,能够在一定概率下检测到敌手恶意行为.公开可验证(public verifiable covert, PVC)方案在满足隐蔽安全条件的基础上,参与方生成公开可验证的欺骗证据,该模型适用于恶意行为被发现会遭受惩罚的现实场景.Covert 模型和 PVC 模型通常只适用于 GC 和 OT 两种 MPC 技术.

MPC 的安全目标包括参与方数据隐私性和计算结果正确性等.进一步,对于参与方获得的计算输出结果,MPC 协议能够达到的安全属性由强到弱依次为:保证结果输出(guaranteed output delivery, GOD)、公平性(fairness, FN)、一致中止(unanimious abort, UA)和选择中止(selective abort, SA).假设敌手可以控制 t 个参与方,诚实方占多数(honest majority)时门限值满足 $t<n/2$,能够达到 GOD,FN 安全性;不诚实方占多数(dishonest majority)时门限值满足 $t\geqslant n/2$,包括 2 个参与方中有一个不诚实参与方的情况,能够达到 UA,SA 安全性.

2 MPC 技术发展

2.1 混淆电路

混淆电路(GC)技术最初关注两方参与的技术方案,技术发展分别从性能提升和安全性提升 2 方面展开,安全性主要体现在半诚实安全与恶意安全的区别,在具有同等安全的情况下,提升方案性能是解决 GC 实用性的重要研究方向.此外,随着 MPC 实际应用需求的增加,近年多方参与的 GC 方案研究也逐渐增多.本节将对两方参与的半诚实安全方案、两方参与的恶意安全方案及多方参与的方案分别进行介绍.图 1 展示了两方参与的 GC 技术发展脉络图.

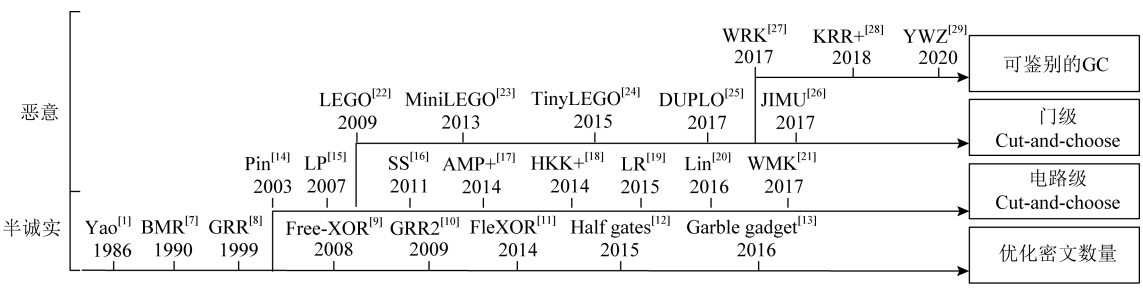

Fig. 1 Technological development of 2-party garbled circuit

图 1 两方参与的混淆电路技术发展脉络

2.1.1 半诚实安全两方混淆电路的性能优化

混淆电路是姚期智于 1986 年提出的一种电路加密技术,参与方混淆器(Garbler)和评估器(Evaluator)对各自的隐私输入计算目标函数,首先需要将目标函数转换成布尔电路的形式,布尔电路由多个二输入单输出的电路门级联而成.GC 的构造是从单个电路门开始,先加密一个门再延伸到加密整个电路.GC 工作流程如图 2 所示,以 AND 门为例,设电路门导线的索引为 a,b 和 c,混淆器拥有导线 a 的输入值 v_a,评估器拥有导线 b 的输入值 v_b.首先,在 GC 生成阶段,混淆器为所有导线选取随机数作为标签 L_{w,v_w},下标 w 为导线索引,v_w 为导线取值,取值为 0 或 1.按照真值表顺序依次使用输入标签加密输出标签得到 4 个密文,打乱密文排列顺序得到混淆表.然后,双方交互传输信息:混淆器向评估器发送混淆表、混淆器的输入标签 L_{a,v_a};评估器通过和混淆器执行 OT 获得自己的输入标签 L_{b,v_b}.最后,在 GC 评估阶段,评估器使用输入标签解密混淆表

密文获得输出标签,当电路门是输出门时,评估器使用输出标签解密输出门密文得到明文输出.对于每个电路门,混淆器生成 6 个随机数、使用两重对称加密生成 4 个密文,评估器依次对 4 个密文解密直到解密成功为止.在整个过程中,GC 的通信量取决于交互阶段双方传输的信息,包括混淆表、混淆器标签和 OT 传输评估器标签,主要的通信开销是混淆表.GC 的计算开销来源于混淆器生成混淆表密文、评估器解密混淆表密文时的计算.

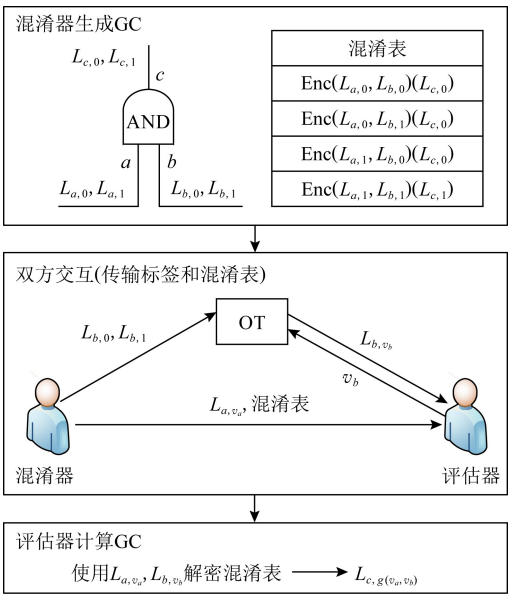

Fig. 2 The diagram of garbled circuit

图 2 混淆电路工作流程

混淆电路提出至今,通信开销一直是其性能瓶颈,半诚实安全模型下的很多方案都致力于通过减少单个门的密文数量来降低通信开销.

Beaver 等人[7]于 1990 年提出的 BMR 方案构造了 point-and-permute(p&p)方法,混淆器为输入导线 w 选取掩码比特 λ_w,对导线真实值 v_w 和掩码比特计算异或得到排列比特 p_{w,v_w},即 $p_{w,v_w}=v_w\oplus\lambda_w$.混淆器在生成混淆表时,按照 2 个输入导线的排列比特对密文进行排列,并将排列比特和标签一起发给评估器.评估器解密时无需依次尝试 4 个密文,只需解密排列比特位置的密文即可获得输出标签,并且由于掩码比特是随机的,评估器不能根据排列比特推测出混淆器的真实输入,混淆电路的隐私性也得到了保证.

Naor 等人[8]于 1999 年提出了 GRR(garbled row reduction, GRR),将单个电路门的密文数量从 4 个减少到 3 个.混淆器生成混淆表时,令混淆表中的第一个密文为 0.若评估器要解密第一个密文,则对密文 0 解密获得输出导线标签.2009 年 Pinkas 等人[10]提出的 GRR2,进一步将密文数量降为 2 个,然而该方案使用多项式插值的方法实现,需要计算模幂运算,导致计算效率低.

Kolesnikov 等人[9]于 2008 年提出了 Free-XOR,其计算 XOR 门的通信开销为 0 个密文,是目前计算 XOR 门的最优方案.混淆器选取全局随机数 R,在生成 GC 时令每根导线 2 个标签取值相差固定值 R,即 $L_{w,1}=L_{w,0}\oplus R$.评估器计算 XOR 门时将两方的输入标签异或得到输出标签,无需计算和传输混淆表,但是计算 AND 门时仍需要混淆表,因此 Free-XOR 适用于 XOR 门较多的运算.Kolesnikov 等人[11]于 2014 年提出了 FleXOR,计算 AND 门的方法与 GRR2 兼容,只需要 2 个密文,计算 XOR 门需要 0～2 个密文.

Zahur 等人[12]于 2015 年提出的半门(Half gates)是目前计算 AND 门的最优方案,该方案与 Free-XOR 兼容,计算 AND,XOR 门的通信开销分别为 2,0 个密文.Half gates 将 AND 门转换为 2 个半门的异或,即 $c=a\wedge b=a\wedge(r\oplus r\oplus b)=(a\wedge r)\oplus(a\wedge(r\oplus b))$.混淆器半门计算 $a\wedge r$,评估器半门计算 $a\wedge(r\oplus b)$,其中,混淆器拥有导线 a 的输入值 v_a,评估器拥有导线 b 的输入值 v_b,令 $r=\lambda_b$(λ_b 为混淆器已知的掩码比特).在 GC 生成阶段,混淆器为混淆器半门生成 2 个密文 $H(L_{a,0})\oplus L_{GC,0}$,$H(L_{a,1})\oplus L_{GC,0}\oplus\lambda_b R$,对 2 个密文进行 p&p 和 GRR 处理后,混淆器半门的密文减少为 1 个;混淆器为评估器半门生成 2 个密文 $H(L_{b,\lambda_b})\oplus L_{EC,0}$,$H(L_{b,\lambda_b\oplus 1})\oplus L_{EC,0}\oplus L_{a,0}$,对 2 个密文做 GRR 处理后评估器半门的密文减少为 1 个,该半门无需做 p&p 处理,因为将评估器输入看作 $v_b\oplus\lambda_b$,那么评估器拥有的 v_b 则为排列比特.评估器获得混淆表和 2 个半门的输入标签 L_{a,v_a},$L_{b,v_b\oplus\lambda_b}$ 后,分别解密 2 个密文获得混淆器半门、评估器半门的输出标签 L_{GC},L_{EC},将 2 个输出标签异或得到 AND 门的输出标签.

Ball 等人[13]于 2016 年提出 Garbled gadgets 可用于混淆算术电路和布尔电路.算术电路的实现是将 Free-XOR 从模 2 扩展到模 m,此时每个导线 w 具有 m 个标签且相邻标签之间相差全局值 R,该方案实现了无通信开销的模 m 的加法和常数乘法.

混淆电路的性能优化不仅包括对通信性能的优化,也有一些方案对于计算性能进行优化.早期 GC

通过对输入标签计算哈希、将哈希结果与输出标签异或来生成混淆表密文.英特尔内核的专用 AES-NI 指令用硬件加速了 AES 运算,激励 Bellare 等人[30]于 2013 年提出使用固定密钥 AES 实现的 GC 方案,比使用哈希算法实现的 GC 方案快 50 倍.此后大多数 GC,OT 方案都使用固定密钥 AES 实现,然而 2020 年 Guo 等人[31]发现很多 GC 方案在实现时因错误使用固定密钥 AES 而易受攻击,为此提出了安全使用固定密钥 AES 的方法并给出安全证明.

2.1.2 恶意安全两方混淆电路的性能优化

半诚实安全模型往往不能满足现实生活中的需求,恶意混淆器会做出发送错误的电路给评估器或通过错误地执行协议来推测评估器的输入等恶意行为.为此,一些抵御恶意敌手的混淆电路方案被提出,包括基于 Cut-and-choose 的混淆电路、可鉴别的混淆电路(Authenticated-GC)等.基于 Cut-and-choose 的混淆电路的性能优化目标是在满足特定安全条件时最小化传输电路的数量,通常只支持两方计算.Authenticated-GC 将秘密分享技术应用于混淆电路中,并为秘密份额添加消息鉴别码,均衡了通信开销和交互轮数.

1) 基于 Cut-and-choose 的混淆电路

使用零知识证明技术来保证参与方正确执行协议会带来巨大开销.为了降低开销,Pinkas[14]于 2003 年首次将 Cut-and-choose 技术引入到 GC 中,Lindell 等人[15]于 2007 年提出了第一个具有完整安全性证明的基于 Cut-and-choose 的恶意安全 GC.在基于 Cut-and-choose 的 GC 中,混淆器生成 σ(σ 为复制因子)个 GC 并发送给评估器,评估器随机选择 c 个电路作为检测电路,剩余电路作为评估电路,恶意敌手攻击成功的概率为 $2^{-\rho}$(ρ 为统计安全参数).评估器要求混淆器提供生成检测电路的随机数,来验证检测电路是否实现约定函数,检测通过后,按正常混淆电路的流程对评估电路进行计算以确定最终计算结果.

在基于 Cut-and-choose 的 GC 方案中,需要解决的问题是输入一致性问题(input consistency)和选择失败攻击(selective failure attack)问题.输入一致问题指的是恶意混淆器向评估器提供的 σ 个混淆电路的输入不一致,影响最终计算结果.选择失败攻击指的是恶意混淆器将电路门混淆表的其中一个密文篡改为错误的密文,如果这个密文刚好是评估器需要解密的密文,评估器在解密时检测到错误会退出,此时混淆器便会知道评估器的排列比特,进一步使用自己拥有的评估器的掩码比特计算出评估器的真实输入.

对于输入一致性问题,早期的工作[15]使用承诺方案来解决,通信开销较高.Shelat 等人[16]于 2011 年提出一致性检测方法,通过传输少量参数代替零知识证明来降低通信开销.Lindell[20]于 2016 年提出输入恢复(input recovery)技术,当混淆器输入不一致导致评估器得到不同输出结果时,可以惩罚混淆器提供真实输入.Wang 等人[21]于 2017 年提出在输出导线中编码陷门的方法,使得评估器在获得多个输出时可以恢复出混淆器的输入.此外,还有一些方案[17,32-33]也给出了解决输入一致性问题的办法.

对于选择失败攻击问题,文献[15]中评估器将真实输入与多个随机比特的异或值作为 OT 输入,使得评估器退出时与真实输入无关,混淆器无法通过选择失败攻击推测评估器输入.Shelat 等人[34]通过构造探测矩阵降低了使用 OT 的数量,然而导致异或运算次数为评估器输入大小的平方,Wang 等人[21]通过将评估器的输入分为多个块并构造更小的探测矩阵来减少计算开销.

基于 Cut-and-choose 的 GC 性能提升问题也受到广泛关注.Yan 等人[18]于 2014 年提出了批处理(Batched)Cut-and-choose,两方对同一函数 f(不同输入数据)进行 N 次安全计算时,批处理 Cut-and-choose 方案中的混淆器生成 $N\rho+c$ 个电路,评估器随机选择 c 个电路作为检测电路,将剩余的电路随机分配到 N 个 bucket 中,每个 bucket 中包含计算一次 f 所需的电路,减少了每次计算的摊销成本.Lindell 等人进一步对该技术进行了改进和实现[19,35].先前单一 Cut-and-choose 的复制因子为 $O(\rho)$,而批处理 Cut-and-choose 的复制因子为 $2+O(\rho/\log N)$,可见其极大地降低了开销.

由于将 Cut-and-choose 应用于整个电路开销很大,于是 Nielsen 等人[22]2009 年首次提出应用于电路门的 Cut-and-choose 方案 LEGO,该方案中混淆器构造大量 NAND 门,双方对这些电路门执行批处理 Cut-and-choose,评估器随机选择部分电路打开检测正确性,将剩余门随机分配给 bucket 并集成到混淆电路中.评估器在评估电路时,取出 bucket 中大多数输出作为该门的输出.LEGO 使用了 Pederson 同态承诺,导致效率较低,Frederiksen 等人[23]于 2013 年提出的 MiniLEGO 采用基于 OT 的异或同态承诺协议,Frederiksen 等人[24]于 2015 年提出的 TinyLEGO 通过优化 bucket 的构造来提高

通信效率.Kolesnikov 等人[25]于 2017 年提出的 DUPLO 可以对任意电路组件进行 Cut-and-choose,电路组件的大小范围可以是单个电路门到整个电路.Zhu 等人[26]于 2017 年提出的 JIMU 通过使用异或同态哈希来代替先前的异或同态承诺.

公开可验证(PVC)安全最早于 2012 年提出,阿里巴巴[36]于 2019 年首次实现了公开可验证的混淆电路,该方案使用 Cut-and-choose 的方法来实现,混淆器发送多个电路给评估器,评估器选择一个电路用于评估,剩余电路用于检测.每个参与方的所有行为都自动带有类似签名的机制以供其他参与方存证,当发现恶意行为时将其签名公开,令恶意参与方承受名誉损失,以此来威慑理性的参与方正确执行协议.

2) 可鉴别的混淆电路

先前的方案均存在一定局限性:对整个电路实施 Cut-and-choose 的通信、计算开销大;对电路门实施 Cut-and-choose 的运行时间长;恶意安全秘密分享协议的交互轮数多等.针对现有协议的不足,Wang 等人[27]于 2017 年提出恒定轮数的可鉴别混淆电路 Authenticated-GC.为了解决选择失败攻击问题,该方案引入预处理程序将导线掩码比特拆分为 2 个份额,将其分别分发给混淆器和评估器,然而掩码比特拆分后导致混淆器无法独立地生成混淆表,需要混淆器和评估器这 2 个参与方分布式地生成混淆表.由于混淆器可能构造错误的混淆表份额影响最终计算结果,该方案引入 BDOZ 型 MAC[37]使得双方可鉴别对方混淆表份额的正确性.

Authenticated-GC 方案中混淆器和评估器协同生成混淆电路,预处理程序包括"与计算函数无关的预处理"和"与计算函数有关的预处理"."与计算函数无关的预处理"为混淆器和评估器生成全局密钥、输入导线掩码比特的份额及对应的 MAC,混淆器和评估器可以在本地计算出 XOR 门的输出导线掩码比特及 MAC.然而计算 AND 门时,需要"与计算函数有关的预处理"生成 AND 三元组,即预处理程序根据从混淆器和评估器收到的输入导线掩码比特计算输出导线的掩码比特,并返回给混淆器和评估器,然后两方使用各自的份额生成混淆表份额.在数据输入阶段双方对彼此的掩码比特鉴别通过后再执行 OT 传输标签,在电路评估阶段评估器逐层使用输入导线标签计算输出导线标签及鉴别信息,在电路输出阶段评估器鉴别输出导线 MAC 正确后使用标签解密 GC 获得结果.

Katz 等人[28]于 2018 年提出安全两方计算场景下 Authenticated-GC 的优化方案,通过设计与半门方案兼容的可鉴别混淆电路、避免每次混淆都发送 MAC 以及优化 AND 门的三元组来提高计算和通信效率.

2.1.3 多方参与的混淆电路

1) 基于 BMR 的多方混淆电路

为了能够让多个参与方协同地生成混淆电路,Beaver 等人[7]于 1990 年提出 BMR 协议,将姚氏混淆电路扩展到多方场景,各参与方为每条导线生成加密值,对所有参与方生成的同一导线的加密值计算异或,得到该导线最终的加密值.

原始 BMR 方案只能抵御半诚实行为敌手,但与其他安全计算技术(SPDZ,SHE 和 OT)合用可以抵御恶意行为敌手.最初 BMR 使用零知识证明技术来证明生成导线加密值时正确使用了伪随机函数(pseudo random function, PRF),计算开销较大.SPDZ-BMR[38], SHE-BMR[39] 和 OT-BMR[40] 对 PRF 的证明和检查有所不同:SPDZ-BMR 在评估电路时使用 SPDZ 型 MAC 来检查 PRF 值和密钥;SHE-BMR 使用 SHE 来验证 PRF 值的正确性;OT-BMR 使用 COT 来对 PRF 值进行一致性检查.SPDZ-BMR,SHE-BMR 和 OT-BMR 中每个门的通信开销依次为 $O(n^4\kappa)$,$O(n^3\kappa)$,$O(n^2B^2\kappa)$,其中 $B=O(1+\rho/\log|C|)$.可见 2017 年提出的 OT-BMR 为这几个方案中性能最优的方案.

2017 年 Wang 等人[41]将可鉴别混淆电路方案与 BMR 技术结合,设计了多方混淆电路方案,通信开销为 $O(n^2B\kappa)$.Yang 等人[29]于 2020 年进一步提出优化方案,针对文献[27]中"与计算函数有关的预处理"进行改进,优化了 AND 三元组生成方式,并首次提出将半门方案应用于多方场景,其中三方计算"与计算函数有关的预处理"的通信效率提高了 35%.

2) 三方混淆电路

实际应用中也存在少数参与方计算混淆电路的场景,例如只有 3 个参与方.安全三方计算通过秘密分享实现的方案较多,混淆电路也有少量专门支持 3 个参与方的方案,通常包含 2 个混淆器以及 1 个评估器,评估器通过 2 个混淆器生成的电路一致性来判断是否有恶意行为.CKM+14[42]将 Cut-and-choose 应用于三方混淆电路,评估器 P_3 的输入通过 XOR 拆分为多个份额,混淆器 P_1,P_2 基于 BMR 协作生成 GC,评估器 P_3 分别和 P_1,P_2 通过 Cut-and-choose 的方式验证双方生成 GC 的正确性.

MRZ15[43]需要2个混淆器协商一致的随机数来生成混淆电路,能够容忍一个恶意混淆器,评估器采用秘密分享拆分其输入并从2个混淆器得到输入份额的标签,进一步恢复出真实输入的标签,该方案仅使用对称加密而无需使用OT,从而提高了计算效率.MRZ15之后被应用于混合计算框架ABY³,可以与三方秘密分享技术进行转换.

2.2 不经意传输

不经意传输(OT)是密码学的基本协议原语之一,由Rabin于1981年提出.OT技术主要包括OT,OT扩展以及相关不经意传输(COT).OT在MPC场景中有广泛的应用:OT自身可实现多项式计算[44-45]和比较大小[46];OT作为重要的MPC安全组件可与其他MPC技术结合应用,例如OT在混淆电路中承担着传输标签的任务;COT能够为秘密分享方案生成相关随机数[47]、消息鉴别码等.

Even等人[48]于1982年提出1-out-of-2 OT的概念,如图3(a)所示.发送方Alice输入2个消息m_0,m_1,接收方Bob输入选择比特b,OT向Bob输出m_b,协议执行过程中Alice不能获知选择比特b且Bob不能获知m_{1-b}.OT可以由基于不同安全假设的公钥加密算法来实现,包括基于DDH实现的NP01[49]、基于LWE实现的BD18[50]和MR19[51]等.

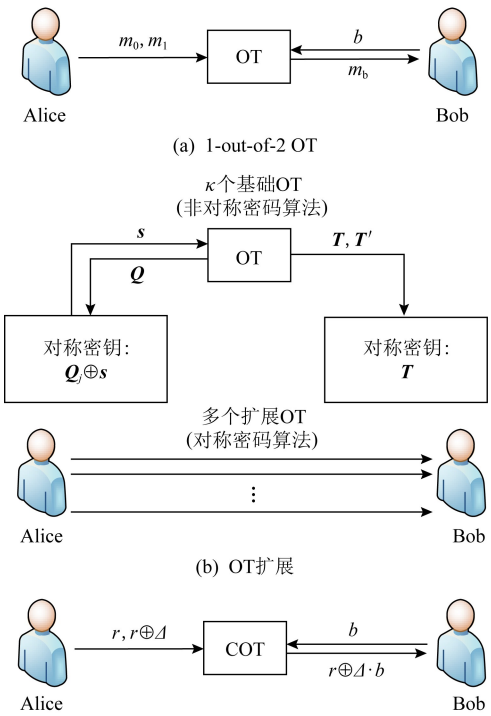

Fig. 3 The technic of oblivious transfer

图3 OT技术原理

然而使用非对称密码算法生成大量的OT会带来很大的计算开销,并且Impagliazzo等人[52]于1989年证明无法在不使用公钥密码的情况下实现OT.为了降低计算开销,研究者们提出了OT扩展技术,使用混合加密方式来生成OT,即先使用少量的公钥算法生成对称密钥,再进一步使用对称加密来完成消息的不经意传输.

1) OT扩展

Beaver[53]于1996年首次提出OT扩展,使用κ(κ为计算安全参数)个基础OT(base OT)可以生成κ的多项式个OT,但是该方案需要使用GC计算复杂的伪随机数发生器导致效率很低.

Ishai等人[54]于2003年提出的OT扩展协议IKNP03,是第一个高效地将m个OT转换为用κ个基础OT来实现的工作($m \gg \kappa$),工作流程如图3(b)所示.IKNP03协议中,两方的输入分别为:Alice输入m对比特串$x_j^0, x_j^1 (1 \leqslant j \leqslant m)$,每个比特串长度为$l$位;Bob输入选择向量$\boldsymbol{r}=(r_1, r_2, \cdots, r_m)$.两方的输出分别为:Alice的输出为空,Bob的输出为$(x_1^{r_1}, x_2^{r_2}, \cdots, x_m^{r_m})$.IKNP03的运算分为基础OT和OT扩展2个阶段:

在基础OT阶段,Bob复制κ份选择向量$\boldsymbol{r}$得到$m \times \kappa$维矩阵$\boldsymbol{R}$,然后选取$m \times \kappa$维随机矩阵$\boldsymbol{T}$并计算$\boldsymbol{T}' = \boldsymbol{T} \oplus \boldsymbol{R}$.Alice和Bob互换原本的发送方和接收方身份提供输入:Bob作为发送方输入矩阵$\boldsymbol{T}$,$\boldsymbol{T}'$,Alice作为接收方输入长度为κ位的随机比特串$\boldsymbol{s}$.基础OT执行结束后Alice获得输出矩阵$\boldsymbol{Q}$,该矩阵第i列$(1 \leqslant i \leqslant \kappa)$满足$\boldsymbol{Q}^i = (s^i \cdot \boldsymbol{r}) \oplus \boldsymbol{T}^i$,第$j$行$(1 \leqslant j \leqslant m)$满足$\boldsymbol{Q}_j = (r_j \cdot \boldsymbol{s}) \oplus \boldsymbol{T}_j$.

在OT扩展阶段,由于$\boldsymbol{Q}_j = (r_j \cdot \boldsymbol{s}) \oplus \boldsymbol{T}_j$表示$r_j = 0$时$\boldsymbol{Q}_j = \boldsymbol{T}_j$,$r_j = 1$时$\boldsymbol{Q}_j = \boldsymbol{s} \oplus \boldsymbol{T}_j$,因此Alice分别使用$\boldsymbol{Q}_j$,$\boldsymbol{Q}_j \oplus \boldsymbol{s}$计算消息$x_j^0, x_j^1$的密文并发送给Bob:$y_j^0 = x_j^0 \oplus H(j, \boldsymbol{Q}_j)$,$y_j^1 = x_j^1 \oplus H(j, \boldsymbol{Q}_j \oplus \boldsymbol{s})$.最后,Bob使用$\boldsymbol{T}$解密获得$x_j^{r_j} = y_j^{r_j} \oplus H(j, \boldsymbol{T}_j)$.

恶意接收方Bob可以在基础OT阶段使用不同的选择向量来同时获得发送方Alice的2个消息,因此需要对接收方的输入进行一致性检测来发现恶意行为.IKNP03在半诚实协议的基础上,引入了Cut-and-choose技术来在OT扩展阶段检测Bob输入的一致性,这种检测方式带来了很大的开销.

研究者们提出了一系列半诚实模型下OT扩展的性能优化方案.Asharov等人[55]于2013年提出的方案ALSZ13分别设计了相关不经意传输(COT)扩展和随机不经意传输(Random OT, ROT)扩展.

COT 中发送方拥有的输入消息是相关的，例如 Free-XOR 的标签传输；ROT 中发送方拥有的消息不相关的.Kolesnikov 等人[56]于 2013 年提出的方案 KK13 适用于传输短秘密值，该应用场景包括 GMW 的 AND 运算等.KK13 改变了 IKNP03 中选择向量 $\boldsymbol{r}$ 的重复编码方式，Bob 对 $\boldsymbol{r}$ 采用随机编码后可在 OT 扩展阶段获得 1-out-of-n OT.当传输的短消息长度为 1 位时 KK13 比 IKNP03 的通信效率提高 $O(\log\kappa)$倍.由于 KK13 方案里 n 取值上限为 256，Kolesnikov 等人[57]提出方案 KKRT16 对其进行改进，Bob 通过使用 PRF 对 $\boldsymbol{r}$ 编码，实现了 1-out-of-∞ OT 扩展.Boyle 等人[58]于 2019 年提出的方案 BCG + 19b 基于伪随机关系随机数生成器(pseudorandom correlation generator, PCG)实现了静默 OT 扩展，使用 PCG 实现静默预处理时，参与方之间只需要少量通信来生成随机数种子，然后在本地对短随机数种子进行扩展来生成具有关系的长随机数.BCG+19b 中生成一个 OT 平均只需要 0～3 位的通信开销，极大地降低了 OT 扩展的通信量，但是其轮数复杂度高，适用于通信带宽资源受限的场景.

此外，还有一些方案用来提高恶意安全模型下 OT 扩展的效率.由于基于 Cut-and-choose 实现的一致性检测方案[59-60]性能太差，Nielsen 等人[61]于 2012 年提出另一种一致性检测的方法 NNOB12，使用哈希函数在基础 OT 阶段对 Bob 的选择向量 $\boldsymbol{r}$ 进行一致性检测，实现该方案需要 2.7κ 个基础 OT.ALSZ15[62]在 NNOB12 的基础上进行改进，将基础 OT 数量减少为 $\kappa+\rho$.KOS15[63]在 IKNP03 半诚实安全方案的基础上改进协议，在 OT 扩展阶段对接收方输入向量进行一致性检测.在 KOS15 中，Alice 和 Bob 协商一个行数索引集 C，接收方 Bob 计算 $\boldsymbol{T}^*=\oplus_{j\in C}\boldsymbol{T}_j$ 和 $\boldsymbol{R}^*=\oplus_{j\in C}\boldsymbol{R}_j$ 并发送给发送方 Alice，然后 Alice 计算 $\boldsymbol{T}^*\oplus(\boldsymbol{R}^*\cdot\boldsymbol{s})$并检查 $\boldsymbol{Q}^*=\boldsymbol{T}^*\oplus(\boldsymbol{R}^*\cdot\boldsymbol{s})$是否成立，若不成立则退出协议.KOS15 仅需 κ 个基础 OT，与 IKNP03 半诚实安全方案相比并未引入额外的开销.3 个方案 NNOB12，ALSZ15 和 KOS15 都实现了恶意安全的 1-out-of-2 的 OT 扩展，OOS16[64]方案在 KK13 的基础上加入与 KOS15 类似的一致性检测获得恶意安全的 1-out-of-n OT 扩展($n\leqslant 2^{76}$)，该协议只需要 κ 个基础 OT，运行时间比 KK13 增长约 20%.

2) 相关不经意传输

相关不经意传输(COT)最早在 ALSZ13 中提出，其功能是生成相关的随机数.COT 的流程如图 3(c)所示，发送方的输入为 r，$r\oplus\Delta$，接收方的输入索引为 b，COT 向接收方输出 $r\oplus\Delta b$.向量不经意线性评估(vector oblivious linear-function evaluation, VOLE)是 COT 的广义定义，接收方可以获得发送方所持有信息的线性组合.

KOS15 提出可以使用 COT 来实现 OT 扩展中的基础 OT 协议.Boyle 等人[65]于 2018 年提出的方案 BCGI18 基于 LPN 假设构造了半诚实安全的 VOLE，通过 PCG 在本地生成关系向量.Boyle 等人[66]于 2019 年提出的 BCG+19a 生成分布式密钥时使用可穿孔的伪随机函数(puncturable pseudorandom function, PPRF)代替 BCGI18 中的分布式点函数(distributed point function, DPF)，并且通过安全设置 PPRF 密钥将协议的安全性提升为恶意安全性，该方案的通信开销比 IKNP03 减少 1 000 倍，计算效率只提高了 6 倍.Schoppmann 等人[67]的并行工作 SGRR19 基于 primal-LPN 实现了半诚实安全的 VOLE 协议，通信效率只比 IKNP03 提高 20 倍，计算效率高于 BCG+19a.Ferret20[68]基于 SGRR19 改进了 VOLE 方案，实现的半诚实安全方案将通信开销减少 15 倍.在半诚实方案的基础上引入一致性检测实现恶意安全性，平均每个 COT 的运行时间比半诚实安全方案慢 1～3 ns.

以上 COT 方案中，BCG+19a 和 Ferret 基于 VOLE 实现了 OT 扩展，而 SGRR19 没有实现 OT 扩展.

表 1 对 OT 扩展方案从安全模型、轮数、通信量和运行时间等方面进行对比.其中，κ 取值为 128，N 为比特向量位数，通信量、运行时间为生成 1 个 OT 时的平均通信量、平均运行时间.通过对比得出 BCG+19a 为目前通信效率最好的方案，Ferret20 为目前计算效率最好的方案.

Table 1 Comparison of OT Extension Schemes

表 1 OT 扩展技术性能对比

方案	恶意安全模型	轮数	通信量/b	运行时间/ns
Bea96[53]	否	2	κ 的多项式	
IKNP03[54]	否	3	128	
KK13[56]	否	3	80	
KOS15[63]	是	3	128	
BCG+19b[58]	否	$\log N$	0～3	
BCG+19a[66]	是	2	0.1	209
Ferret20[68]	是	2,4	0.44	18

2.3 秘密分享

传统的秘密分享方案中,秘密分发者将秘密拆分为份额并分发给参与方,只有足够数量的秘密份额组合在一起才能够恢复出秘密信息.基于秘密分享的安全多方计算(secret sharing-secure multiparty computation, SS-MPC)指的是参与方对不同秘密数据的份额进行运算,得到计算结果的份额,足够数量的结果份额组合在一起能够恢复出计算结果.本节围绕 SS-MPC 来展开介绍,而不关注 SS 技术本身.

SS-MPC 根据秘密的生成方式可以分为基于多项式插值的秘密分享和加性秘密分享.基于多项式插值的 SS-MPC 容易实现任意门限,但是需要模幂运算,导致其计算效率低.加性秘密分享根据秘密数据形式可以分为算术秘密分享和布尔秘密分享,在计算线性运算时通常需要使用算术秘密分享,在计算非线性运算时通常需要使用布尔秘密分享.加性秘密分享的优势是计算效率高,其门限方案的实现需要参与方持有冗余份额.目前 SS-MPC 在实际中应用最广泛,能够高效地实现线性运算和非线性运算.

2.3.1 SS-MPC 计算线性运算

SS-MPC 根据系统中敌手控制的参与方数量,可分为诚实方占多数的 SS-MPC 和不诚实方占多数的 SS-MPC.

1) 诚实方占多数

SS-MPC 在早期有一些经典方案.GMW 方案[69]于 1987 年被提出,能够支持多个参与方进行布尔运算,秘密数据以异或的形式拆分,计算 XOR 运算可在本地完成,计算 AND 运算需要参与方两两之间执行 1-out-of-4 OT,导致交互轮数较多.BGW 方案[70]于 1988 年被提出,是基于 Shamir 秘密分享实现的 SS-MPC 方案,但其计算乘法要用到随机化和降阶处理,导致开销较大.Beaver 三元组[71](Beaver triples)于 1991 年被提出,是 SS-MPC 中最常用的乘法组件.该方案在预处理阶段选取随机数 a,b 和 c,满足 $c=ab$,并为其生成加性份额 a_i,b_i 和 c_i($1\leqslant i\leqslant n$).在线阶段,参与方 P_i 拥有秘密数据 x,y 的加性份额 x_i,y_i,P_i 在本地计算 x_i-a_i,y_i-b_i 并广播;收到其他参与方广播的份额后 P_i 将份额相加得到 $x-a$,$y-b$,然后对已有数据计算得到积的份额 $z_i=(x-a)b_i+(y-b)a_i+c_i$.秘密恢复阶段,每个参与方提供 z_i,对 z_i 求和再与 $(x-a)(y-b)$ 相加便能恢复出 xy.

近年来由于机器学习等应用场景的需要,涌现出基于 SS-MPC 实现的 PPML 方案,包括 ABY^3 和 SecureNN 等.这些方案的底层协议是加法、乘法和比较等基本算子的 SS-MPC 协议,高层协议实现了机器学习模块的隐私保护,本节主要关注这些方案的底层协议.由于加法运算在本地对份额相加即可,乘法运算的流程相对复杂,因此主要对乘法运算进行分析.

SS-MPC 最初提出时交互轮数多、通信量也较大,大多数 SS-MPC 方案都是对这 2 个指标进行优化.优化方法主要包括引入相关随机数、将计算前移到预处理阶段以及增加计算方的数量等.为了解决计算方的单点失效问题,一些方案构造了门限秘密分享来提高系统容错能力.

2008 年提出的 Sharemind[72]设计了半诚实安全的三方 SS-MPC 方案.秘密分享的过程为:数据拥有方对数据 x,y 计算加性份额 x_i,y_i($i=1,2,3$)并分发给计算方 P_i.计算乘法的过程为:P_i 在本地计算 x_iy_i,3 个计算方交互计算得到交叉项 x_iy_j(i,$j=1,2,3$).例如计算交叉项 x_1y_2,P_3 选取随机数 a_1,a_2 并分别分发给 P_1,P_2,然后计算 $w_3=a_1a_2$ 作为自己的份额;P_1 和 P_2 分别计算 x_1+a_1,y_2+a_2 并发送给对方,然后 P_1,P_2 通过计算分别得到份额 $w_1=-a_1(y_2+a_2)$,$w_2=y_2(x_1+a_1)$.至此,每个 P_i 分别得到交叉项 x_1y_2 的份额 w_i,满足 $w_1+w_2+w_3=x_1y_2$,其他交叉项的计算同理.最终 P_i 将自己的本地份额与交叉项份额相加得到 xy 的份额.

为了降低 Sharemind 的轮数和通信量,2016 年 Araki 等人提出的方案 AFL+16[73]引入相关随机数,并通过设置冗余秘密份额构造了半诚实安全的 2-out-of-3 SS-MPC 方案.该方案设计了算术、布尔 2 种秘密分享方案,以算术秘密分享为例,秘密分享的过程为:数据拥有方选取 3 个和为 0 的 l 位长随机串 a_1,a_2,a_3,满足 $a_1+a_2+a_3=0$,然后计算秘密数据 x 的加性份额 $x_i=a_{i-1}-x$($i=1,2,3$),令 $i=1$ 时 $i-1=3$,同理可计算秘密数据 y 的加性份额.数据拥有方向计算方 P_i 分发份额 (a_i,x_i),(b_i,y_i).计算乘法的过程为:P_i 选取随机数 α_i,计算 $r_i=(x_iy_i-a_ib_i+\alpha_i)/3$ 并将 r_i 发给 P_{i+1},其中 $\alpha_1+\alpha_2+\alpha_3=0$;收到其他参与方的信息后,$P_i$ 计算积的份额 (c_i,z_i):$c_i=-r_{i-1}-r_i$,$z_i=-2r_{i-1}-r_i$.秘密恢复时任意 2 个计算方可以恢复出 xy.

2018 年提出的 ABY^3[74]受到 AFL+16 设置冗余份额的启发,基于此设计了新的 2-out-of-3 SS-MPC 方案,进一步通过份额校验实现了恶意安全

方案.秘密分享的过程为:数据拥有方计算长度为 l 位的秘密数据 x,y 的加性份额 x_i,y_i($i=1,2,3$),并向 3 个计算方 P_i 分发份额(x_i,x_{i+1}),(y_i,y_{i+1}).计算乘法的过程为:P_i 在本地计算得到积的份额 $z_i=x_iy_i+x_iy_{i+1}+x_{i+1}y_i+\alpha_i$ 并将其发给 P_{i-1},其中 $\alpha_1+\alpha_2+\alpha_3=0$.在秘密恢复阶段,任意 2 个计算方可以恢复出 xy. ABY^3 通过使用 Beaver 三元组来完成份额校验,实现的恶意安全方案最多能够容忍一个恶意参与方.

为了降低 ABY^3 在线阶段的通信量和交互轮数,2019 年提出的 ASTRA[75]将一个计算方的运算前移到预处理阶段进行.秘密分享的过程为:在离线阶段数据拥有方 P_0 同时也是计算方,分别与计算方 P_1,P_2 使用 PRF 生成 2 个随机数 $r_{x,1}$,$r_{x,2}$,在线阶段 P_0 计算秘密数据 x 的份额 $m_x=x+r_x$ 并发送给 P_1,P_2,其中 $r_x=r_{x,1}+r_{x,2}$,P_1,P_2 分别设置 x 的份额为(m_x,$r_{x,1}$),(m_x,$r_{x,2}$),秘密 y 的份额同理可得.计算乘法的过程为:离线阶段 P_0 和 P_1 生成随机数 $r_{z,1}$,$r_{xy,1}$,P_0 和 P_2 生成随机数 $r_{z,2}$,$r_{xy,2}$,满足 $r_{xy,1}+r_{xy,2}=r_xr_y$.在线阶段的计算过程中,2 个计算方 P_i($i=1,2$)分别计算 $m_{z,i}=(i-1)\times m_xm_y-m_xr_{y,i}-m_yr_{x,i}+r_{z,i}+r_{xy,i}$,双方交换份额相加得到 $m_z=xy+r_{z,1}+r_{z,2}$,设置份额为(m_z,$r_{z,i}$).在秘密恢复阶段,任意 2 个计算方可以恢复出 $xy=m_z-r_{z,1}-r_{z,2}$.ASTRA 的恶意安全方案在半诚实安全方案的基础上加入一致性检测,在输入和输出阶段通过对比份额的哈希值来检查份额一致性,在计算阶段使用 Beaver 三元组验证正确性.

2020 年提出的 BLAZE[76]对 ASTRA 中一致性检测的预处理过程进行改进,将其通信量减少 1/7.同年,TRIDENT[77]用另一种思路来提升 ASTRA 的性能,设计了四方秘密分享方案,在线阶段需要 3 个参与方两两之间会生成积的份额及其哈希值,并发给另一方检验一致性.

SecureNN[78]是 2019 年提出的半诚实安全三方及四方 SS-MPC 方案.数据拥有方生成 $m\times n$ 维秘密矩阵 $\boldsymbol{x}$,$n\times v$ 维秘密矩阵 $\boldsymbol{y}$ 的加性份额 $\boldsymbol{x}_i$,$\boldsymbol{y}_i$($i=1,2$),并分发给计算方 P_1,P_2,矩阵元素是长度为 l 位的比特串.三方秘密分享方案中计算乘法时采用 Beaver 三元组的思想,P_0 作为辅助节点来生成 Beaver 三元组以及其他随机数,P_1 和 P_2 是计算节点.四方秘密分享方案中 P_1,P_2 是计算节点,P_3,P_4 是辅助计算节点.计算乘法运算时首先 P_1,P_2 分别在本地计算 $\boldsymbol{x}_1\boldsymbol{y}_1$,$\boldsymbol{x}_2\boldsymbol{y}_2$;然后,$P_1$ 分别把 $\boldsymbol{x}_1$,$\boldsymbol{y}_1$ 发给 P_3,P_4,P_2 分别把 $\boldsymbol{y}_2$,$\boldsymbol{x}_2$ 发给 P_3,P_4;收到份额后,P_3,P_4 分别计算交叉项 $\boldsymbol{x}_1\boldsymbol{y}_2$,$\boldsymbol{x}_2\boldsymbol{y}_1$ 并发送给 P_1,P_2.在秘密恢复阶段,将 P_1 和 P_2 拥有的份额相加即可恢复出矩阵 $\boldsymbol{xy}$.

2) 不诚实方占多数

在恶意安全的 SS-MPC 方案中,可以通过向秘密份额添加消息鉴别码,来检测计算过程中计算方是否存在篡改行为.这类方案主要包括 BDOZ 和 SPDZ 等,能够用于不诚实方占多数的场景,达到了中止安全性,即发现参与方恶意行为时中止协议.

Bendlin 等人[35]于 2011 年提出了 BDOZ 消息鉴别码,在 2 个参与方的场景下,P_1,P_2 分别拥有全局密钥 Δ_1,Δ_2,以及秘密 x 的加性份额 x_1,x_2.秘密 x 的 BDOZ 份额用$[x]$来表示,P_1 拥有$[x]_1=(x_1,M_1,K_1)$,P_2 拥有$[x]_2=(x_2,M_2,K_2)$.其中 K_1,K_2 是 MAC 密钥,M_1,M_2 是 MAC 值,满足关系 $M_1=K_2+\Delta_2x_1$,$M_2=K_1+\Delta_1x_2$.鉴别 P_1 的份额 x_1 时,P_1 向 P_2 提供(x_1,M_1),P_2 使用(K_2,Δ_2)验证等式 $M_1=K_2+\Delta_2x_1$ 是否成立,若等式成立则鉴别通过,鉴别 P_2 的份额同理.BDOZ 可以扩展到 n 个参与方的场景,但是每个参与方都需要对其他 $n-1$ 个参与方的 MAC 密钥分别生成一个 MAC,存储开销和参与方的数量成线性关系.

为了解决 BDOZ 中的存储开销问题,Damgård 等人[79]于 2012 年提出另一种消息鉴别码 SPDZ,每个参与方只存储恒定数量的 MAC.SPDZ 型 MAC 表示为 $M(x)=\alpha\times x$,其中 α 为 MAC 全局密钥、x 为秘密数据.在 SPDZ 方案中,每个参与方 P_i($1\leqslant i\leqslant n$)持有密钥 α 的加性秘密份额 α_i,x 的加性秘密份额 x_i 和 MAC 值 $M(x)$的加性秘密份额 $M(x)_i$.验证 MAC 时需要先恢复出 x,α 和 $M(x)$再检查等式 $M(x)=\alpha\times x$ 是否成立.SPDZ 型 MAC 满足加法、常数乘法的同态性,计算乘法时需 Beaver 三元组协助来验证份额.具体实现时,SPDZ 在预处理阶段使用 FHE 生成 Beaver 三元组和 MAC,并使用零知识证明保证 FHE 密文的正确性.

Damgård 等人于 2013 年提出 SPDZ 的改进方案[80],该方案在验证 MAC 时无需恢复 MAC 密钥,因此可以继续对还未验证的秘密数据进行计算.验证 MAC 时各参与方首先广播 x_i,收到广播的份额后计算得到 x,然后各参与方分别计算 $M_i-x\alpha_i$,对 n 个参与方的 $M_i-x\alpha_i$ 求和,若求和的结果为 0 则验证通过.

Keller 等人[47]于 2016 年提出 MASCOT,保持 SPDZ 的在线阶段不变,在预处理阶段使用 COT 生成 Beaver 三元组.Keller 等人[81]于 2018 年提出方案 KPD2018,使用 BGV 的加法同态代替原始方案中的 FHE 来生成 Beaver 三元组,获得了比 MASCOT 更好的性能.

Cramer 等人[82]于 2018 年提出了有限环上的方案 $SPDZ_{2K}$,环上的计算更接近于 CPU 计算单元的实际计算情况,有限环上的安全计算具备很高的实际意义,该方案的效率和有限域上的方案基本一致.

2.3.2 SS-MPC 计算非线性运算

SS-MPC 实现的非线性运算包括比较运算和最高有效位(most significant bit, MSB)运算.

ABY^3 方案计算线性运算采用算术秘密分享,计算非线性运算采用布尔秘密分享,算术份额转换为布尔份额需要做比特分解运算,即将长度为 l 位的算术份额 x 转换为布尔份额 $x[1],x[2],\cdots,x[l]\in\{0,1\}$,份额满足 $x=\sum_{i=1}^{l}2^{i-1}x[i]$. 布尔份额在秘密恢复时需要计算比特间的加法,考虑低位向高位的进位问题,使用并行前缀加法器(parallel prefix adder, PPA)实现.比较秘密值 x 和常数 c 的大小时,可以转换为计算 $x-c$ 的最高有效位.参与方先计算 $x-c$ 的算术份额,然后对计算结果进行比特分解得到 $x-c$ 的布尔份额,最后使用 PPA 恢复出最高有效位即可得到 x 和 c 的大小关系.ABY^3 实现安全比较运算需要交互 $\log l+1$ 轮,通信量为 $9l$ 位.

为了降低轮数复杂度和通信量,ASTRA 未使用 PPA,而是只使用秘密分享来完成最高有效位的计算,只需要 3 轮交互和 $4(l+2)$ 位的通信量即可完成运算.根据前文介绍的 ASTRA 的秘密分享方式,P_1,P_2 分别拥有秘密 x 的份额 $(m_x,r_{x,1})$,$(m_x,r_{x,2})$,P_0 拥有份额 r_x,其中 $r_x=r_{x,1}+r_{x,2}$,$m_x=x+r_x$.计算最高有效位时,离线阶段 P_1,P_2 选取长度为 l 位的随机数 r,r' 并令 $s=MSB(r)$.当 2 个数用二进制补码的形式表示时,计算乘法时积的正负等于 2 个乘数正负的异或,即 $MSB(rx)=MSB(r)\oplus MSB(x)$.因此,在线阶段的计算过程中,首先 P_1 计算 $x_1=m_x-r_{x,1}$ 并将 rx 的份额 rx_1+r' 发送给 P_0,P_2 计算 $x_2=-r_{x,2}$ 并将 rx 的份额 rx_2-r' 发送给 P_0.然后 P_0 将份额相加得到 rx,计算最高有效位 $s'=MSB(rx)$,然后计算 s' 的份额 s'_1,s'_2,并分别发送给 P_1,P_2.最后参与方 $P_i(i=1,2)$ 计算 $MSB(x)_i=s_i\oplus s'_i$.秘密恢复阶段,任意 2 个参与方都能够恢复出最高有效位 $MSB(x)$.

SecureNN 实现的比较运算可以比较秘密值和常数值的大小,参与方拥有 $\mathbb{Z}_p$ 上秘密值 x 的比特份额和明文比特串 r,秘密值和常数值长度均为 l 位.从低位到高位依次为第 $1,2,\cdots,l$ 个比特计算比较运算时,对 x 和 r 的第 i 位从高位到低位依次计算 $w_i=x[i]\oplus r[i]=x[i]+r[i]-2x[i]r[i]$,$c_i=r[i]-x[i]+1+\sum_{k=i+1}^{l}w_k$. 由于 $r[i]-x[i]+1\geqslant 0$,所以只要存在 $c_i=0$ 则说明同时满足 $\sum_{k=i+1}^{l}w_k=0$ 和 $r[i]-x[i]=-1$,即 x 和 r 的第 $i+1$ 位至 l 位都相等且第 i 位 $x[i]>r[i]$,由此可得 $x>r$;若不存在 $c_i=0$ 则 $x<r$. SecureNN 计算 MSB 运算是通过将 $\mathbb{Z}_{l-1}$ 上 a 的最高有效位计算转换为 $\mathbb{Z}_l$ 上 $2a$ 的最低有效位计算,需要调用比较运算来实现.

表 2 对 SS-MPC 方案的安全模型和性能进行了对比.我们对乘法、比较(或最高有效位)运算的

Table 2 Comparison of SS-MPC Schemes

表 2 SS-MPC 安全模型和性能比较

方案	参与方	门限方案	安全模型	乘法		比较/最高有效位	
				通信量/b	轮数	通信量/b	轮数
Sharemind[72]	3	否	半诚实	$5l$	3		
AFL+16[73]	3	否	半诚实	$3l$	1		
ABY3[74]	3	是	半诚实	$3l$	1	$9l$	$1+\log l$
			恶意	$9l$	1	$18l$	$1+\log l$
ASTRA[75]	3	是	半诚实	$2l$	1	$4l+2$	3
			恶意	$4l$	1	$8l+1$	4
BLAZE[76]	3	是	恶意	$3l$	1	$9l$	$1+\log l$
TRIDENT[77]	4	是	恶意	$3l$	1	$5l+2$	3
SecureNN[78]	3	否	半诚实	$2(2mn+2nv+mv)l$	2	$4l\log p+16l+2$	5
	4	否	半诚实	$2(2mn+nv+mv)l$	2	$4l\log p+2l+4$	6

通信量和交互轮数2个方面进行比较,其中 l 为秘密比特串长度,p 为有限域的大小.通过对比发现,ASTRA是半诚实安全模型下性能最好的方案,TRIDENT和BLAZE在恶意安全模型下实现的乘法运算具有较好的性能,但是在计算最高有效位时TRIDENT的性能更好.SecureNN的乘法性能为矩阵乘法运算的性能,其他方案是单个数据乘法运算的性能.

2.4 同态加密

同态加密支持对多个数据的密文进行运算,解密得到的结果与数据明文运算结果一致.即明文与密文的运算满足同态性质:

$$Dec(Enc(x+y))=Dec(Enc(x)\oplus Enc(y)),$$

其中,+和⊕分别是明文和密文空间上的运算.

早在1978年Rivest等人就提出了隐私同态(privacy homomorphism)的概念,但这个概念一直被认为是一个开放性的问题,直到2009年Gentry提出首个全同态加密方案,证明了在加密数据上计算任何函数的可行性.

同态加密可以划分为部分同态加密(partial homomorphic encryption, PHE)、浅同态加密(somewhat homomorphic encryption, SHE)和全同态加密(fully homomorphic encryption, FHE).部分同态加密仅支持单一类型的密文同态运算,主要包括加法同态(additive homomorphic encryption, AHE)和乘法同态(multiplicative homomorphic encryption, MHE),代表方案分别为Paillier[83]和ElGamal[84].浅同态加密支持低次多项式运算.全同态加密的构造均遵循Gentry的思想蓝图,利用自举(bootstrapping)技术将浅同态加密方案转换为全同态加密方案,即通过同态地执行解密电路以更新密文、约减噪音,从而支持进一步的同态运算.目前主流的FHE方案大多基于格上的困难问题(主要是LWE, RLWE)构造,代表方案包括BGV[85], BFV[86-87], GSW[88], CGGI[89], CKKS[90]等.

基于(R)LWE构造的同态方案中,密钥切换(key switching)技术和模切换(modulus switching)技术分别用于解决同态乘法运算导致的密文维数和噪音量级增长问题.为了降低自举导致的性能开销,BGV方案利用密钥切换和模切换技术实现了一种不需要自举就能构造的层次全同态加密(leveled fully homomorphic encryption, leveled FHE)方案.层次全同态加密根据方案预期计算的电路深度设置参数,能够计算任意多项式深度的电路,已经可以满足多数应用场景下的计算需求.BFV方案中提出了一种新的张量技术,构造了一个标量不变的leveled FHE方案.GSW方案是基于近似特征向量法建立的简单全同态加密方案,消除了密钥切换过程,使得同态加法和乘法在很大程度上就是矩阵的加法和乘法.CGGI方案通过在环面上构造RGSW和RLWE的外部积,对加密比特进行门自举,在运行时间和可用性上具有优势.CKKS方案构造类似于BGV方案,支持定点数的近似运算.

根据明文空间的不同,同态加密方案可以进一步分为逐字同态加密(word-wise HE)和逐比特同态加密(bit-wise HE)两类.逐字运算的HE方案例如BGV, BFV和CKKS等,优点是支持打包技术,可以将多个明文打包为单个密文并以单指令多数据方式对明文进行操作,提高了运算效率;早期的逐字运算方案只支持加法及乘法等多项式计算,难以有效支持除法、根号等非多项式运算,2020年Cheon等人[91]提出通过复合多项式近似符号函数使用CKKS来实现同态比较.逐比特运算的HE方案包括TFHE, FHEW[92]等,优点是可以支持非多项式运算,但是由于不能使用打包技术导致性能显著低于逐字运算方案.阿里巴巴于2020年提出的全同态加密技术PEGASUS有效桥接逐字运算方案CKKS和逐比特运算方案FHEW两类全同态加密技术,使用CKKS密文有效计算线性函数,转换为FHEW密文形式后可通过查表来计算非线性运算.

单密钥同态方案只能对同一个密钥加密的密文进行运算,而多密钥全同态加密(Multikey-FHE, MKHE)支持对不同密钥加密的密文进行同态运算,但需要参与方联合解密密文.2012年首个NTRU型的MKHE方案提出[93],此后MKHE技术迅速发展,相继提出GSW型的MKHE[94-97]、BGV型的MKHE[98-99]和TFHE型的MKHE[100].MKHE方案的构造和实现上仍需进一步的优化,主要关注多跳功能(允许运算过程中有新的参与方加入)的实现、密钥(参与方)数量上限、密文增长、密钥切换优化等多个方面.

全同态加密可用于机器学习中的隐私保护并保持非交互性,比如PEGASUS[101]可以使用2种同态技术实现机器学习中的线性和非线性运算.文献[100]中数据拥有方和模型拥有方分别使用自己的公钥加密数据,并将其发送给服务器,服务器能够对2种密钥加密的密文进行运算.

3 MPC 基础技术的比较

本节对混淆电路、秘密分享和同态加密 3 类 MPC 基础技术的计算方数量、计算量、通信量和交互轮数进行分析对比,如表 3 所示:

Table 3 Comparison of MPC Fundamental Schemes

表 3 不同 MPC 基础技术对比

技术	计算方数量	计算量	通信量	交互轮数
GC	2,n	中	大	恒定
SS	>2	小	中	线性
HE	2,n	大	中	恒定

GC-MPC 由于实现了底层的基本逻辑运算符,因此具有通用性,可面向不同的应用逻辑.GC-MPC 一般有 2 个计算方,若是基于 BMR 实现的协议则可以支持多个计算方.混淆电路的计算开销主要是对称加密或哈希函数的计算,计算开销较小;混淆电路交互轮数是恒定的 2 轮,通信量与电路门的数量成正比.GC-MPC 在计算简单逻辑运算时具有优势,如比较运算和加法运算,但是在计算复杂运算时会产生庞大的电路从而导致通信量很大、性能降低.目前没有出现针对大数据量复杂计算的 GC-MPC 处理平台,GC-MPC 只在一些小数据量、简单计算场景中有具体应用,比如拍卖、薪水公平性调研等.

SS-MPC 的计算方数量一般大于 2,SS-MPC 计算方的运算都是简单的加法和乘法运算,不涉及模幂运算,因此计算开销小.SS-MPC 的轮数和电路深度成线性关系,导致计算方端到端通信会有较长时延.近年来随着算法优化和网络带宽不断提升,SS-MPC 的性能优势不断凸显,逐渐成为目前应用最广的 MPC 技术.SS-MPC 不仅能高效计算加法和乘法等线性运算,还能够计算比较大小、最高有效位和除法等非线性运算.目前 SS-MPC 的实际应用有 Sharemind MPC 等通用计算平台,以及 CrypTFlow 等 PPML 开源实现.

HE-MPC 通常支持两方运算,多密钥 HE 支持多方运算.HE-MPC 的轮数恒定、通信量比前 2 种 MPC 技术小.但是 HE-MPC 的计算开销很大,其中同态加法和标量乘法(明文和同态密文相乘)计算开销相对较小,然而同态乘法无论是自举,还是 leveled FHE 都需要对密文进行降维和降噪从而产生大量计算开销.此外,一些 HE 方案密钥和密文空间复杂度高,需要较大的存储开销.对于一些具有深度较大的布尔或算术电路,同态加密计算效率难以被实际应用接受.

在实际应用中,计算函数一般是线性运算和非线性运算的组合,由于 GC,SS 和 HE 适用于不同场景,将多种 MPC 技术结合的混合技术能够获得更好的性能.表 4 中对现有 MPC 的实现进行总结.

Table 4 The Implementations of MPC Schemes

表 4 MPC 方案方现

技术	方案实现
GC	EMP-toolkit[102], Obliv-C[103], ObliVM[104], CBMC-GC[105], Frigate[106]
OT	EMP-toolkit[102], LibOTe①
SS	MP-SPDZ[107], Wysteria[108], PICCO[109], FRESCO②, JIFF③, MPyc④
HE	SEAL⑤, HElib⑥, PALISADE⑦, HEAAN⑧, FHEW⑨, TFHE⑩, Lattigo⑪, cuFHE⑫, FV-NFLib⑬
Hybrid	ABY[110], ABY^3[74], SCALE-MAMBA⑭

4 隐私保护机器学习

机器学习广泛应用于计算机视觉、语音识别和自然语言处理等领域.图 4 为机器学习层次结构图,第 1 层为机器学习典型算法,第 2 层为机器学习组成模块,第 3 层为机器学习基本算子.

机器学习算法主要包括神经网络(neural networks, NN)、线性回归(linear regression)和逻辑回归(logistic regression)等.神经网络是机器学习应用最广泛的算法,由输入层、隐藏层和输出层构成,其中隐藏层包括卷积层(convolutional layer, CONV)、全连接层(fully connected layers, FC)和

① https://github.com/osu-crypto/libOTe
② https://github.com/aicis/fresco
③ https://github.com/multiparty/jiff
④ https://github.com/lschoe/mpyc
⑤ https://github.com/microsoft/SEAL
⑥ https://github.com/homenc/HElib
⑦ https://git.njit.edu/palisad/PALISADE
⑧ https://github.com/snucrypto/HEAAN
⑨ https://github.com/lducas/FHEW
⑩ https://github.com/tfhe/tfhe
⑪ https://github.com/ldsec/lattigo
⑫ https://github.com/vernamlab/cuFHE
⑬ https://github.com/CryptoExperts/FV-NFLlib
⑭ https://github.com/KULeuven-COSIC/SCALE-MAMBA

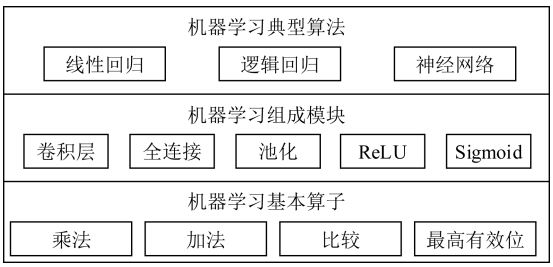

Fig. 4 The hierarchy of machine learning

图 4 机器学习层次结构图

池化层(pooling layer).卷积层的运算是过滤器矩阵和数据特征向量的内积运算;全连接层的运算是先计算神经元向量与权重矩阵的内积再与偏移向量相加;池化层是计算过滤器窗口内元素的最大值或平均值,分别称为最大池化层和平均池化层.神经网络隐藏层和输出层都要使用激活函数(activation function)为神经元引入非线性因素,使得神经网络可以逼近任何非线性函数,隐藏层常用的激活函数有 *ReLU*,*Sigmoid* 和 tanh.线性回归和逻辑回归算法相对简单,都可以看作是单层神经网络,其中逻辑回归使用了 *Sigmoid* 激活函数.

从机器学习底层的基本算子来看,卷积层、全连接层和平均池化层可以使用加法和乘法 2 种线性运算实现.最大池化层可以使用比较大小或最高有效位等非线性运算来实现.激活函数在 PPML 方案中通常被转换为线性运算和非线性运算的组合,例如将 *ReLU* 表示为 $ReLU(x)=\max(0,x)=(MSB(x)\oplus 1)\times x$,文献[111]中提出可以使用多项式来近似激活函数 $Sigmoid(x)=1/(1+e^{-x})$:

$$Sigmoid(x)=\begin{cases}0, & x<(-1/2),\\ x+(1/2), & (-1/2)\leqslant x\leqslant(1/2),\\ 1, & x>(1/2).\end{cases}$$

将激活函数转换为加法、乘法和比较等基本算子组合函数的方法,可以更好地使用 MPC 来进行运算.

神经网络模型训练最常用的优化算法是梯度下降优化(gradient descent optimization)算法,包括正向传播和反向传播 2 个过程:正向传播是输入信息通过输入层经隐藏层逐层处理并传向输出层,然后对输出值计算损失函数;反向传播是依据微积分中的链式法则沿着输出层到输入层计算中间变量和参数的梯度,并更新参数;训练模型需要重复正向传播和反向传播多次直到模型收敛.神经网络推理的运算是一次正向传播的过程.在机器学习的计算过程中,用户数据和模型参数是浮点数的形式,而 MPC 只能对整数类型的数据做运算,因此在使用 MPC 完成 PPML 时,要将浮点数转换为固定点数(固定精度的整数),并且在计算乘法后要计算截断运算.

机器学习分为 3 种明文架构,在客户端-服务器架构中,客户端向服务器发送明文数据,服务器计算推理结果并返回给客户端.在外包计算架构中,数据拥有方(模型拥有方)将其数据(模型)发送给一组外包服务器,服务器得到运算结果后将其返回给数据拥有方.在联邦学习架构中,客户端在本地使用私有数据训练得到局部模型并将其上传到中央服务器,服务器聚合得到更新的模型.

针对明文的机器学习架构无法保证用户私有数据和服务商模型参数的隐私性,在明文架构中引入 MPC 技术能够实现隐私保护,得到 3 种基于 MPC 的 PPML 架构,分别为安全客户端-服务器架构(secure client server, S-CS)、安全外包计算(secure outsourced computation, S-OC)、安全联邦学习(secure federated learning, S-FL),如图 5 所示:

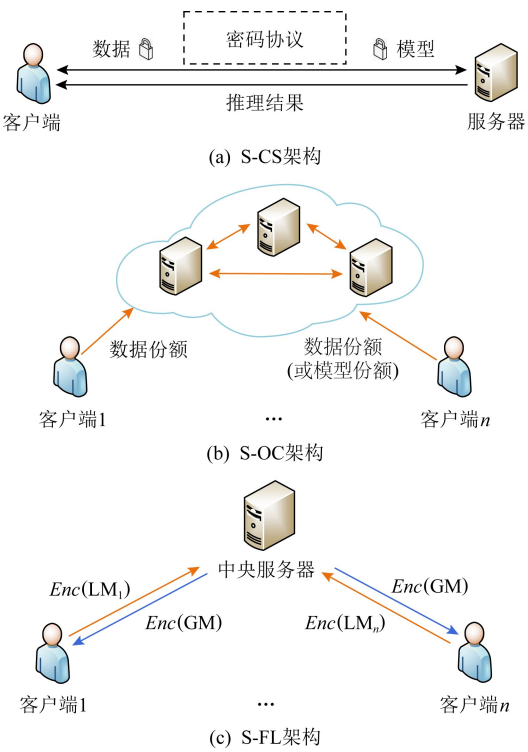

Fig. 5 The construction of PPLM based MPC

图 5 基于 MPC 的 PPML 架构图

基于 S-CS 架构的 PPML 方案用于做神经网络推理,允许客户端在不向服务器透露其私有数据的情况下获得神经网络推理结果,同时保护服务器模

型参数的隐私.该架构有时需要客户端和服务器协同计算,由于服务器拥有更强的算力,在设计算法时倾向于为服务器分配相对复杂的运算.与明文 CS 架构相比,S-CS 架构中客户端通常需要参与运算.

基于 S-OC 架构的 PPML 方案支持神经网络的训练和推理.数据拥有方对私有数据计算份额后发送给一组不会合谋的服务器.在神经网络推理时,模型参数拥有方以秘密分享的方式将其机器学习模型托管到一组服务器上,客户端将私有数据秘密分享给这组服务器,每个服务器对私有数据份额、模型参数份额进行模型推理的协同计算,每个服务器将得到的推理结果份额返回给客户端,客户端对其进行秘密恢复后可得到完整的推理结果.在神经网络训练时,多个数据拥有方以秘密分享的方式将数据集托管到一组服务器上,服务器在联合数据集上训练机器学习模型.与明文 OC 架构不同,S-OC 架构的计算服务器彼此之间不能合谋.

基于 S-FL 架构的 PPML 方案可以防止中央服务器执行模型逆向攻击(model inversion attack)来推测出用户的私有训练集,与明文 FL 架构不同,S-FL 架构客户端在本地得到局部模型参数后,需要对其加密后再上传至中央服务器.

4.1 基于 S-CS 架构的 PPML

从 MPC 技术、性能和准确率 3 个方面对基于 S-CS 架构的 PPML 进行总结,如表 5 所示.表 5 中的方案都实现了多个基准工作的评估,本节选取部分评估实验来对比性能.其中 Delphi 和 CrypTFlow2 使用深度神经网络 ResNet-32 对 CIFAR-100 数据集进行推理,其余方案只在浅层神经网络实现了对 MNIST 数据集的推理.

Table 5 The PPML Schemes based on S-CS Architecture

表 5 基于 S-CS 架构的 PPML 方案

神经网络类型	方案	MPC 技术		性能		准确率/%
		线性层	非线性层	协议运行时间/s	通信量/MB	
NN	CryptoNets[112](ICML'16)	leveled FHE	leveled FHE	297.50	372.20	98.95
	MiniONN[113](CCS'17)	AHE+SS	GC,SS	1.28	47.60	99.00
	Chameleon[114](ASIACCS'18)	SS	GC,SS	2.24	10.50	99.00
	GAZELLE[115](USENIX'18)	AHE	GC	0.29	8.00	
	Delphi[116](USENIX'20)	AHE+SS	GC+SS	6.99		92.00
	CrypTFlow2[46](CCS'20)	AHE,COT	OT	0.63		93.20
BNN	DeepSecure[117](IACR'18)	GC	GC	9.67	791.00	99.00
	XONN[118](USENIX'18)	OT+GC	GC	0.16	38.28	99.00

注:"+"表示需要结合 2 种技术;","表示 2 种技术任选其一.

从实现 S-CS 的 MPC 技术来看,CryptoNets,DeepSecure 分别是基于单一技术 HE 和 GC 设计的方案,分别产生了较大的计算开销和通信开销.之后的 PPML 方案考虑线性层和非线性层的不同计算特性,组合多种 MPC 技术实现安全运算,对此类方案分析为:

1) S-CS 架构的线性层实现.主要使用 AHE 和 SS 技术,标量乘法可以转换为 AHE 运算.MiniONN,GAZELLE 和 Delphi 都使用 AHE 实现了线性层.GAZELLE 与 MiniONN 相比,减少了同态运算次数、客户端和服务器的交互次数,并且设计了适用于机器学习的同态线性代数核.Delphi 与 GAZELLE 相比,将在线阶段繁重的同态运算前移到预处理运算,降低了在线阶段的计算开销.Chameleon 和 CrypTFlow2 分别使用 SS,COT 来实现线性层的运算.此外,DeepSecure 和 XONN 都是应用于模型参数为二进制数的网络,因此使用 GC 来实现线性运算.

2) S-CS 架构的非线性层实现.主要使用 SS,GC 和 OT 技术,使用 SS 来执行非线性层的运算.通常先对非线性运算进行多项式近似,然后再使用 SS 来执行运算,近似运算会导致准确率降低.使用 GC 来执行非线性层的运算虽然可以提高准确率,但是会增加通信开销,例如 GAZELLE.CrypTFlow2 使用 OT 来执行非线性运算,能够同时保证准确率和通信效率,获得了优于 GC 约 7 倍的通信效率.此外,也有一些方案为了同时获得可观的准确率和通信效率,分别使用 GC 和 SS 各完成一部分非线性运算,例如 MiniONN,Chameleon 和 Delphi.

3) S-CS 架构的 MPC 技术演进分析.通用神经网络(不包括 BNN)线性层的实现只包含 HE 和 SS

技术,最早使用 leveled FHE 的方法来实现,其巨大的计算开销严重影响了方案的可用性.此后提出的 AHE 实现的优化包括算法层面的优化,即构造适用于 S-CS 场景的同态线性代数核;以及协议层面的优化,即将繁重的 HE 运算前移到预处理阶段,同时结合 SS 来完成线性运算,降低在线阶段的计算开销.通用神经网络非线性层技术是影响整个方案性能开销、准确率的关键,GC 和 SS 分别只能满足高准确率和高性能,而无法同时达到最优效果.2020 年提出的 CrypTFlow2 使用 OT 计算非线性运算,在保证准确率的前提下极大地提高了计算、通信效率.

CryptoNets[112] 是 Microsoft 提出的第一个使用同态加密实现隐私保护机器学习的方案,该方案的缺点是使用 LHE 友好的激活函数(例如平方函数)代替常用的 $ReLU$ 和 $Sigmoid$,影响了结果准确性,并且在牺牲准确性的前提下,计算成本依然非常大.CryptoNets 每小时可完成 59 000 次推理.

MiniONN[113] 的线性层结合加性秘密分享和同态加密 2 种技术实现,客户端将私有数据拆分为 2 份并将其中一份发给服务器,服务器计算模型参数的同态密文并发给客户端,客户端对该密文和自己的数据份额计算乘法并将结果发给服务器,服务器结合这些信息可以计算出线性层的输出.对于非线性层,使用 GC 来计算激活函数 $ReLU$,使用 SS 来计算激活函数 $Sigmoid$ 和 Tanh 的多项式近似函数.

Chameleon[114] 的线性层使用秘密分享方案 Beaver 三元组来计算,引入可信第三方在离线阶段生成相关随机数.非线性层使用 GMW 或 GC 计算,由于 GMW 的轮数和电路深度有关,因此当电路深度较深时使用 GC 计算非线性层,其余情况使用 GMW 计算非线性层.Chameleon 的运行时间比 MiniONN 快 4.2 倍.

GAZELLE 方案[115] 设计了同态线性代数核来优化同态运算,并使用优化的 AHE 方案计算线性层,使用 GC 来计算 $ReLU$ 所需的比较运算,并通过加性秘密分享技术在 LHE 和 GC 之间有效切换.计算线性层时,客户端计算输入数据 x 的密文 $Enc(x)$并发送给服务器,服务器对 $Enc(x)$和自己拥有的模型明文参数计算标量乘法得到线性层输出的密文 $Enc(y)$.由于密文 $Enc(y)$不能直接作为 GC 的输入,并且如果将 $Enc(y)$发给客户端解密可能会泄露模型参数.使用秘密分享可以解决这个问题:服务器选取随机数作为份额 $-y_s$,并计算 $Enc(y)+Enc(y_s)=Enc(y+y_s)$发送给客户端,客户端解密获得份额 $y_c=y+y_s$,将 y_c 和 $-y_s$ 作为 GC 的输入便能计算非线性函数.GAZELLE 方案在线阶段的运行时间比 Chameleon 快 20 倍.

Delphi[116] 将 GAZELLE 线性层繁重的同态加密运算前移到预处理阶段,客户端私有数据为 x,服务器拥有模型参数 M.在线性层的预处理阶段,客户端选取随机数 r,对其计算同态密文 $Enc(r)$并发给服务器;服务器选取随机数 s 作为线性层份额并计 算 $Enc(Mr-s)$并发给客户端;客户端解密得到 $Mr-s$ 并将其作为线性层份额.在线性层的在线阶段,客户端计算 $x-r$ 并发送给服务器,服务器设置线性层份额为 $M(x-r)+s$,可见双方的份额之和等于线性计算结果 Mx.对非线性层的改进是使用多项式近似激活函数并用 Beaver 三元组来计算,这种方法在减小开销的同时也会导致准确率降低,考虑到效率和准确率的折中,在保证准确率满足阈值的前提下最大化近似计算激活函数的数量,并分别使用 GC 和 SS 计算 2 部分激活函数.Delphi 获得的准确率与明文模型准确率相差不到 0.04%,运行时间比 GAZELLE 快 22 倍、通信效率提升 9 倍.

先前工作在实现 $ReLU$ 时存在通信开销大或准确率低等问题,CrypTFlow2 使用 OT 实现的比较大小运算避免了先前方案的弊端.参与方 P_0,P_1 在比较长度为 l 位的私有数据 x 和 y 时,可以将其拆分为高位、低位比特串的拼接 $x=x_1\|x_0$,$y=y_1\|y_0$,使用式 $1\{x<y\}=1\{x_1<y_1\}\oplus(1\{x_1=y_1\}\wedge 1\{x_0<y_0\})$可得到 x,y 的大小关系.但是拆分为2 个数据块通信开销依然较大,进一步的改进是将 x,y 分别分解为 q 个数据块得到 2 个树的叶子节点,每个数据块长 $m=l/q$ 位.P_0 和 P_1 通过执行 1-out-of-2^m OT 能够获得每对叶子节点的大小、相等关系,从叶子节点开始逐层递归计算 $\log q$ 次便可得到根节点的值 $1\{x<y\}$.此外,CrypTFlow2 还设计了新的截断运算和正整数除法协议,线性层的实现使用 Delphi 中的 HE 方案或基于 COT 的方案.与 Delphi 相比,CrypTFlow2 非线性层的运行时间快了 22 倍、通信效率提升 9.3 倍.

DeepSecure[117] 和 XONN 适用于二进制神经网络的安全推理.DeepSecure 对 GC 组件进行优化并引入了降低开销的预处理技术,运行时间优于 CryptoNets,但同时也带来了更大的通信开销.XONN[118] 将神经网络中线性层的运算转换为向量点乘运算,神经网络第一层客户端输入数据是整数、神经网络权重是二进制,使用 OT 计算整数和二进制数的向量点乘;神经网络中间层运算数据都是

二进制值 0 和 1,使用同或(XNOR)和加法来计算二进制点乘运算,由于可用 XOR 代替 XNOR 运算,且 GC 已经实现 Free-XOR,因此整个方案的开销较小,性能优于 GAZELLE 方案 7 倍.

4.2 基于 S-OC 架构的 PPML

从参与方数量、机器学习功能、安全模型和 MPC 技术等方面对现有基于 S-OC 架构的 PPML 方案进行总结,如表 6 所示:

Table 6 The PPLM Schemes Based on S-OC Architecture

表 6 基于 S-OC 架构的 PPML 方案

方案	参与方数量	训练	推理	安全模型		MPC 技术		准确率/%
				半诚实	恶意	线性层	非线性层	
SecureML[111](S&P'17)	2	是	是	是		SS	GC	93.40
QUOTIENT[119](CCS'19)	2	是	是	是		SS,COT	GC	99.38
ABY^3[74](CCS'18)	3	是	是	是	是	SS	SS,GC	93.00～99.00
ASTRA[75](IACR'19)	3		是	是	是	SS	SS	
BLAZE[76](NDSS'20)	3	是	是		是	SS	SS,GC	93.20～97.80
TRIDENT[77](NDSS'20)	4	是	是		是	SS	SS,GC	93.00～98.30
SecureNN[78](PETS'19)	3,4	是	是	是		SS	SS	93.40～99.15
CrypTFlow[120](S&P'20)	3		是	是	是	SS	SS	
PrivPy[121]	4	是	是	是		SS	SS	

1) S-OC 架构的 MPC 技术.基于 S-OC 架构的方案主要使用秘密分享技术,通常需要 2～4 个计算方协同计算.ABY^3 等方案通过设置冗余的加性秘密份额实现了门限的效果,可以容忍一个参与方的单点失效,增强了系统的容错能力,其中 ABY^3 实现了中止安全性,ASTRA 等实现了安全性更强的公平性.

2) S-OC 架构的安全模型.部分方案只设计了单一安全模型下的方案,例如半诚实安全模型下的 SecureML、恶意安全模型下的 BLAZE,还有一些方案分别构造了半诚实和恶意 2 种安全模型下的方案,例如 ASTRA 等.

3) S-OC 架构支持的机器学习功能.表 6 中所有方案都能支持神经网络推理,除 ASTRA 和 CrypTFlow 以外其他方案都能支持神经网络训练.

4) S-OC 架构与 S-CS 架构的对比.S-CS 架构中应用了多种 MPC 技术,包括 AHE,OT,GC,SS 等.而在 S-OC 架构中主要使用单一的 SS 技术,因为机器学习在训练模型时需要迭代计算多次,SS 是计算开销最小的 MPC 技术,能够有效降低模型训练时间.现有的 HE 和 OT 方案无法有效支持 S-OC 架构下的安全多方计算.

SecureML 是半诚实安全模型下的两方计算方案.整个训练过程分为离线和在线 2 个阶段,离线阶段利用 HE 或者 OT 生成 Beaver 三元组;在线阶段计算线性层运算时使用秘密分享技术,计算激活函数时使用平方函数做近似并通过 GC 来计算,然而近似运算导致精度降低,只获得了 93.4%的准确率.

与 SecureML 结构类似,QUOTIENT[119] 也是半诚实安全模型下的两方计算方案.该方案将模型参数表示为三元值{−1,0,1}、私有数据表示为整数形式,因此计算方需要对算术份额和布尔份额进行运算,使用 COT 来完成乘法运算,使用 GC 来完成非线性运算.QUOTIENT 的模型推理采用 S-CS 架构,而其他 S-OC 方案的模型推理依然是外包给计算方来完成.QUOTIENT 对 ResNet32 模型进行训练,准确率与明文模型相比只降低了 0.1%～2.17%,与 SecureML 相比运行效率提高了 13 倍.

ABY^3 是首个三方计算场景下的混合协议,由算术秘密分享、布尔秘密分享和姚氏电路 3 种 MPC 技术构成.三方的设置无法使用传统的两方混淆电路方案,只能使用专门为三方设计的混淆电路.该方案还设计了不同 MPC 技术的高效切换协议.ABY^3 使用 MNIST 数据集训练得到的模型准确性为 93%～99%,神经网络、线性回归的训练时间比 SecureML 快 80～55 000 倍.

ASTRA 对 ABY^3 线性计算的改进是将一部分在线阶段的运算前移到预处理阶段,对非线性运算的改进是使用秘密分享代替并行前缀加法器,获得了恒定轮数的效果.ASTRA 只实现了线性回归、逻辑回归等模型的安全推理,而未实现模型训练.BLAZE 与 ASTRA 相比的改进是提供对神经网络

模型的支持,并且增加模型训练的功能.TRIDENT 在 ABY³ 的基础上,将线性层的截断计算和乘法计算合并来避免额外的通信开销,并且优化了 ASTRA 的最高有效位计算方法.

SecureNN 仅使用秘密分享技术构造了神经网络中的线性层和非线性层,四方设置比三方设置获得更好的性能,其对 MNIIST 数据集进行模型训练和推理,获得 93.4%～99.15%的准确率,比明文模型训练开销增加 17～33 倍.

SecureNN 等方案在实现时都需要手动在 MPC 友好的低级语言或开发库上实现 PPML 模型,这种方式对 ML 开发者不够友好.CrypTFlow[120] 编译器能够自动将 TensorFlow 的推理代码转换为多种 MPC 协议,在实现编译器后端 MPC 协议时对 SecureNN 三方计算方案的通信效率进行改进,更便于开发者使用.SecureNN 将矩阵 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 的卷积计算转换为更大的矩阵 $\boldsymbol{x}'$ 和 $\boldsymbol{y}'$ 的乘法,然后使用 Beaver 三元组来计算 $\boldsymbol{x}'$ 和 $\boldsymbol{y}'$ 的积来获得卷积结果,其中变形矩阵 $\boldsymbol{x}'$ 有冗余参数会增大通信开销.CrypTFlow 的改进是先使用 Beaver 三元组计算原始矩阵 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 的积,得到中间结果的份额后,参与方在本地对矩阵进行转换,避免交互过程中传输冗余的信息.CrypTFlow 对 SecureNN 非线性层的改进是消除辅助节点向 2 个计算方发送份额的过程,将 *ReLU* 和最大池化层的通信开销降低了 1/4.最后,CrypTFlow 使用 SGX 技术将 MPC 协议从半诚实安全转换为恶意安全.CrypTFlow 目前只实现了神经网络推理,准确性与明文推理相近,半诚实安全方案和恶意安全方案的运行时间分别为 30 s 和 2 min.

此外,国内提出的方案 PrivPy[121] 由语言前端和计算引擎后端组成,前端提供编程接口和代码优化,后端执行基于秘密分享的隐私保护计算.PrivPy 支持 3 种后端:PrivPy 设计的 2-out-of-4 秘密分享协议后端、SPDZ 和 ABY³.PrivPy 设计的后端需要采用二次秘密分享技术,前 2 个参与方获得私有数据份额后,再计算新的份额发送给后 2 个计算方,通过冗余份额的设置,在秘密恢复阶段只要有 2 个参与方即可恢复出计算结果.PrivPy 实现的模型训练和推理的性能均优于 ABY³.

4.3 基于 S-FL 架构的 PPML

联邦学习(FL)是为了解决数据孤岛问题而提出的机器学习算法,由一个中央服务器和多个客户端组成.客户端在本地使用私有数据训练模型,将模型梯度发送至中央服务器.服务器整合多个客户端的模型梯度得到全局模型.FL 架构仅需要传输模型梯度,客户端的私有数据自始至终没有离开本地,即便如此,研究表明中央服务器可以通过对模型逆向攻击来推测出客户端数据.

为了保护用户数据隐私,Shokri 等人[122] 于 2015 年提出的方案在模型准确性和隐私保护之间进行折中,提出将模型梯度的 1%～10%上传到云服务器的方法.Aono 等人[123] 于 2017 年证明攻击者从部分梯度中也能推测出有用信息,并使用密码学来解决隐私泄露的问题:客户端在本地使用加法同态算法对局部模型梯度加密并将密文上传至中央服务器,服务器对各参与方的密文计算同态加法并将结果返回给客户端,客户端解密后得到更新的模型梯度.谷歌[124] 于 2017 年提出了基于秘密分享实现的联邦学习安全聚合方案,参与方使用秘密分享计算掩码向量,使用掩码向量对局部梯度计算异或加密并上传至服务器.服务器先对收到的数据求和,然后使用秘密恢复技术计算出掩码向量,并将其从求和值中消去,得到全局模型.该方案利用 Shamir 的门限特性,只要用户数量满足门限就消去掩码,解决了实际应用中可能存在的用户中途退出问题.谷歌将该方案应用于 GBoard 输入法中来进行单词预测.

目前已有一些基于 MPC 实现的联邦学习框架投入使用.FATE 项目最初使用加法同态加密技术来构建联邦学习底层安全计算协议,在之后的版本上增加了 SPDZ 秘密分享协议,提供更多样化的 MPC 协议支持.此外,2019 年发布的 PySyft 是用于隐私保护深度学习的开源库,将 SPDZ 等 MPC 技术和差分隐私集成到联邦学习框架并向用户提供应用程序接口.

5 总结与展望

安全多方计算提出的前 20 年一直停留在理论层面的研究,没有应用于实际生活中的方案和系统.2000 年以来,各种 MPC 基础技术在实践中都有了相应的系统实现,例如 Fairplay,Sharemind MPC 和 SEAL 等.在大数据应用的推动下,MPC 基础理论的迭代更新和工程实践的广泛应用体现了其学术和实用价值,MPC 在未来的发展中有巨大的潜力和广阔的发展前景.

PPML 是当前最受关注的 MPC 应用领域,近年来在可用性和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,未来工作可围绕以下3个方面展开.

1) 进一步提升准确率与系统性能.MPC实现的PPML方案开销依然远大于明文模型的运算,准确率与明文模型相比还是会有一定的损失.PPML方案非线性层的实现方式会为准确率带来很大的影响,相对而言GC精度较高但性能开销大,SS性能较好但无法达到GC的精度,现有PPML方案都在效率和准确率之间进行折中.最新提出的使用OT实现非线性运算的方案同时获得了性能和准确率的保证,仅适用于两方参与的场景.因此MPC基础技术的优化将对PPML技术的性能优化起到重要作用.此外,在PPML使用混合技术方案中降低不同技术的转换开销也将进一步提升PPML技术的可用性.

2) 提升系统安全性.现有的很多方案都只满足半诚实安全模型,未来需要进一步加强对恶意敌手的防御.另一方面,早期的恶意安全方案实现了中止安全性,2019年之后的方案都达到了更强的安全属性-公平性,提升系统的安全属性也是今后研究的一个重要方向,未来要实现的安全目标是能够保证结果输出(GOD),并且在性能开销也需要控制在可接受范围内.

3) 降低模型隐私泄露风险.现有的部分S-CS架构下的方案会向客户端泄露过滤器的大小、卷积的步长以及神经网络隐藏层的种类等模型信息.未来仍需进一步探索对模型参数信息更有效的保护方案.

参 考 文 献

[1] A C C. How to generate and exchange secrets [C] //Proc of 27th Annual Symp on Foundations of Computer Science. Piscataway, NJ: IEEE, 1986: 162-167

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

[3] Blakley G R. Safeguarding cryptographic keys [C] //Proc of Managing Requirements Knowledge, International Workshop on IEEE Computer Society. Piscataway, NJ: IEEE, 1979: 313-313

[4] Rabin M O. Transaction protection by beacons [J]. Journal of Computer and System Sciences, 1981, 27(2):256-267

[5] Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms [J]. Foundations of Secure Computation, 1978, 4(11): 169-180

[6] Gentry C. Fully homomorphic encryption using ideal lattices [C] //Proc of the 41st Annual ACM Symp on Theory of Computing. New York: ACM, 2009: 169-178

[7] Beaver D, Micali S, Rogaway P. The round complexity of secure protocols [C] //Proc of the 22nd Annual ACM Symp on Theory of Computing. New York: ACM, 1990: 503-513

[8] Naor M, Pinkas B, Sumner R. Privacy preserving auctions and mechanism design [C] //Proc of the 1st ACM Conf on Electronic Commerce. New York: ACM, 1999: 129-139

[9] Kolesnikov V, Schneider T. Improved garbled circuit: Free XOR gates and applications [C] //Proc of Int Colloquium on Automata, Languages, and Programming. Berlin: Springer, 2008: 486-498

[10] Pinkas B, Schneider T, Smart N P, et al. Secure two-party computation is practical [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2009: 250-267

[11] Kolesnikov V, Mohassel P, Rosulek M. Flexor: Flexible garbling for XOR gates that beats free-xor [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2014: 440-457

[12] Zahur S, Rosulek M, Evans D. Two halves make a whole [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2015: 220-250

[13] Ball M, Malkin T, Rosulek M. Garbling gadgets for boolean and arithmetic circuits [C] //Proc of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 565-577

[14] Pinkas B. Fair secure two-party computation [C] //Proc of Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2003: 87-105

[15] Lindell Y, Pinkas B. An efficient protocol for secure two-party computation in the presence of malicious adversaries [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2007: 52-78

[16] Shelat A, Shen C. Two-output secure computation with malicious adversaries [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2011: 386-405

[17] Afshar A, Mohassel P, Pinkas B, et al. Non-interactive secure computation based on cut-and-choose [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2014: 387-404

[18] Huang Yan, Katz J, Kolesnikov V, et al. Amortizing garbled circuits [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2014: 458-475

[19] Lindell Y, Riva B. Cut-and-choose Yao-based secure computation in the online/offline and batch settings [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2014: 476-494

[20] Lindell Y. Fast cut-and-choose-based protocols for malicious and covert adversaries [J]. Journal of Cryptology, 2016, 29(2): 456-490

[21] Wang Xiao, Malozemoff A J, Katz J. Faster secure two-party computation in the single-execution setting [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2017: 399-424

[22] Nielsen J B, Orlandi C. Lego for two-party secure computation [C] //Proc of Theory of Cryptography Conf. Berlin: Springer, 2009: 368-386

[23] Frederiksen T K, Jakobsen T P, Nielsen J B, et al. MiniLEGO: Efficient secure two-party computation from general assumptions [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2013: 537-556

[24] Frederiksen T K, Jakobsen T P, Nielsen J B, et al. TinyLEGO: An interactive garbling scheme for maliciously secure two-party computation [OL]. 2015 [2021-07-29]. https://eprint.iacr.org/2015/309

[25] Kolesnikov V, Nielsen J B, Rosulek M, et al. DUPLO: Unifying cut-and-choose for garbled circuits [C] //Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 3-20

[26] Zhu Ruiyu, Yan Huang. JIMU: Faster lego-based secure computation using additive homomorphic hashes [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2017: 529-572

[27] Wang Xiao, Ranellucci S, Katz J. Authenticated garbling and efficient maliciously secure two-party computation [C] //Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 21-37

[28] Katz J, Ranellucci S, Rosulek M, et al. Optimizing authenticated garbling for faster secure two-party computation [C] //Proc of Annual Int Cryptology Conf. Berlin: Springer, 2018: 365-391

[29] Yang Kang, Wang Xiao, Zhang Jiang. More efficient MPC from improved triple generation and authenticated garbling [C] //Proc of the 2020 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 1627-1646

[30] Bellare M, Hoang V T, Keelveedhi S, et al. Efficient garbling from a fixed-key blockcipher [C] //Proc of 2013 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2013: 478-492

[31] Guo Chun, Katz J, Wang Xiao, et al. Efficient and secure multiparty computation from fixed-key block ciphers [C] //Proc of 2020 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2020: 825-841

[32] Lindell Y, Pinkas B. Secure two-party computation via cut-and-choose oblivious transfer [J]. Journal of Cryptology, 2012, 25(4): 680-722

[33] Brandão L T A N. Secure two-party computation with reusable bit-commitments, via a cut-and-choose with forge-and-lose technique [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2013: 441-463

[34] Shelat A, Shen C H. Fast two-party secure computation with minimal assumptions [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 523-534

[35] Lindell Y, Riva B. Blazing fast 2pc in the offline/online setting with security for malicious adversaries [C] //Proc of the 22nd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2015: 579-590

[36] Hong Cheng, Katz J, Kolesnikov V, et al. Covert security with public verifiability: Faster, leaner, and simpler [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2019: 97-121

[37] Bendlin R, Damgård I, Orlandi C, et al. Semi-homomorphic encryption and multiparty computation [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2011: 169-188

[38] Lindell Y, Pinkas B, Smart N P, et al. Efficient constant round multi-party computation combining BMR and SPDZ [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2015: 319-338

[39] Lindell Y, Smart N P, Soria-Vazquez E. More efficient constant-round multi-party computation from BMR and SHE [C] //Proc of Theory of Cryptography Conf. Berlin: Springer, 2016: 554-581

[40] Hazay C, Scholl P, Soria-Vazquez E. Low cost constant round MPC combining BMR and oblivious transfer [J]. Journal of Cryptology, 2020, 33(4): 1732-1786

[41] Wang Xiao, Ranellucci S, Katz J. Global-scale secure multiparty computation [C] //Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 39-56

[42] Choi S G, Katz J, Malozemoff A J, et al. Efficient three-party computation fromcut-and-choose [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2014: 513-530

[43] Mohassel P, Rosulek M, Zhang Ye. Fast and secure three-party computation: The garbled circuit approach [C] //Proc of the 22nd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2015: 591-602

[44] Kilian J. Founding crytpography on oblivious transfer [C] //Proc of the 20th Annual ACM Symp on Theory of Computing. New York: ACM, 1988: 20-31

[45] Naor M, Pinkas B. Oblivious polynomial evaluation [J]. SIAM Journal on Computing, 2006, 35(5): 1254-1281

[46] Rathee D, Rathee M, Kumar N, et al. Cryptflow2: practical 2-party secure inference [C] //Proc of the 2020 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 325-342

[47] Keller M, Orsini E, Scholl P. Mascot: faster malicious arithmetic secure computation with oblivious transfer [C] //Proc of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 830-842

[48] Even S, Goidreich O, Lempel A. Arandomized protocol for signing contracts [J]. Communications of ACM, 1985, 28 (6): 637-647

[49] Naor M, Pinkas B. Efficient oblivious transfer protocols [C] //Proc of ACM-SIAM Symp on Discrete Algorithms. New York: ACM, 2001: 448-457

[50] Brakerski Z, Döttling N. Two-message statistically sender-private OT from LWE [C] //Proc of Theory of Cryptography Conf. Berlin: Springer, 2018: 370-390

[51] Mansy D, Rindal P. Endemic oblivious transfer [C] //Proc of the 2019 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 309-326

[52] Impagliazzo R, Rudich S. Limits on the provable consequences of one-way permutations [C] //Proc of the 21st Annual ACM Symp on Theory of Computing. New York: ACM, 1989: 44-61

[53] Beaver D. Correlated pseudorandomness and the complexity of private computations [C] //Proc of the 28th Annual ACM Symp on Theory of Computing. New York: ACM, 1996: 479-488

[54] Ishai Y, Kilian J, Nissim K, et al. Extending oblivious transfers efficiently [C] //Proc of Annual Int Cryptology Conf. Berlin: Springer, 2003: 145-161

[55] Asharov G, Lindell Y, Schneider T, et al. More efficient oblivious transfer and extensions for faster secure computation [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 535-548

[56] Kolesnikov V, Kumaresan R. Improved OT extension for transferring short secrets [C] //Proc of Annual Cryptology Conf. New York: ACM, 2013: 54-70

[57] Kolesnikov V, Kumaresan R, Rosulek M, et al. Efficient batched oblivioussprf with applications to private set intersection [C] //Proc of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 818-829

[58] Boyle E, Couteau G, Gilboa N, et al. Efficient pseudorandom correlation generators: Silent OT extension and more [C] //Proc of Annual Int Cryptology Conf. Berlin: Springer, 2019: 489-518

[59] Nielsen J B. Extendingoblivious transfers efficiently-how to get robustness almost for free [OL]. 2007 [2021-07-29]. https://eprint.iacr.org/2007/215.pdf

[60] Harnik D, Ishai Y, Kushilevitz E, et al. OT-combiners via secure computation [C] //Proc of Theory of Cryptography Conf. Berlin: Springer, 2008: 393-411

[61] Nielsen J B, Nordholt P S, Orlandi C, et al. A new approach to practical active-secure two-party computation [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2012: 681-700

[62] Asharov G, Lindell Y, Schneider T, et al. More efficient oblivious transfer extensions with security for malicious adversaries [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2015: 673-701

[63] Keller M, Orsini E, Scholl P. Actively secure OT extension with optimal overhead [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2015: 724-741

[64] Orrù M, Orsini E, Scholl P. Actively secure 1-out-of-N OT extension with application to private set intersection [C] // Proc of Cryptographers' Track at the RSA Conf. Berlin: Springer, 2017: 381-396

[65] Boyle E, Couteau G, Gilboa N, et al. Compressing vector OLE [C] //Proc of the 2018 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2018: 896-912

[66] Boyle E, Couteau G, Gilboa N, et al. Efficient two-round OT extension and silent non-interactive secure computation [C] //Proc of the 2019 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 291-308

[67] Schoppmann P, Gascón A, Reichert L, et al. Distributed vector-OLE: Improved constructions and implementation [C] //Proc of the 2019 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 1055-1072

[68] Yang Kang, Weng Chenkai, Lan Xiao, et al. Ferret: Fast extension for correlated OT with small communication [C] // Proc of of the 2020 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 1607-1626

[69] Micali S, Goldreich O, Wigderson A. How to play any mental game [C] //Proc of the 19th ACM Symp on Theory of Computing. New York: ACM, 1987: 218-229

[70] Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation [C] //Proc of the 20th Annual ACM Symp on Theory of Computing. New York: ACM, 1988: 1-10

[71] Beaver D. Efficient multiparty protocols using circuit randomization [C] //Proc of Annual Int Cryptology Conf. Berlin: Springer, 1991: 420-432

[72] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations [C] //Proc of European Symp on Research in Computer Security. Berlin: Springer, 2008: 192-206

[73] Araki T, Furukawa J, Lindell Y, et al. High-throughput semi-honest secure three-party computation with an honest majority [C] //Proc of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 805-817

[74] Mohassel P, Rindal P. ABY^3: 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

[75] 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

[76] Patra A, Suresh A. BlAZE: Blazing fast privacy-preserving machine learning [C/OL] //Proc of the 27th Network and Distributed System Security Symp. Reston VA: The Internet Society, 2020 [2021-07-29]. https://eprint.iacr.org/2020/042.pdf

[77] Rachuri R, Suresh A. TRIDENT: efficient 4pc framework for privacy preserving machine learning [C/OL] //Proc of the 27th Network and Distributed System Security Symp. Reston VA: The Internet Society, 2020 [2021-07-29]. https://eprint.iacr.org/2019/1315.pdf

[78] 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

[79] Damgård I, Pastro V, Smart N, et al. Multiparty computation from somewhat homomorphic encryption [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2012: 643-662

[80] Damgård I, Keller M, Larraia E, et al. Practical covertly secure MPC for dishonest majority-or: Breaking the SPDZ limits [C] //Proc of European Symp on Research in Computer Security. Berlin: Springer, 2013: 1-18

[81] 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

[82] Cramer R, Damgård I, Escudero D, et al. $SPDZ_{2^k}$: Efficient MPC mod 2^k for dishonest majority [C] //Proc of Annual Int Cryptology Conf. Berlin: Springer, 2018: 769-798

[83] Paillier P. Public-key cryptosystems based on composite degree residuosity classes [C] //Proc of Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 1999: 223-238

[84] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms [J]. IEEE Transactions on Information Theory, 1985, 31(4): 469-472

[85] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) Fully homomorphic encryption without bootstrapping [J]. ACM Transactions on Computation Theory, 2014, 6(3): 1-36

[86] Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2012: 868-886

[87] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption [OL]. 2012 [2021-07-29]. https://eprint.iacr.org/2012/144

[88] Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2013: 75-92

[89] Chillotti I, Gama N, Georgieva M, et al. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2016: 3-33

[90] Cheon J H, Kim A, Kim M, et al. Homomorphic encryption for arithmetic of approximate numbers [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2017: 409-437

[91] Cheon J H, Kim D, Kim D. Efficient homomorphic comparison methods with optimal complexity [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2020: 221-256

[92] Ducas L, Micciancio D. Fhew: Bootstrapping homomorphic encryption in less than a second [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2015: 617-640

[93] Asharov G, Jain A, López-Alt A, et al. Multiparty computation with low communication, computation and interaction via threshold FHE [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2012: 483-501

[94] Clear M, McGoldrick C. Multi-identity and multi-key leveled FHE from learning with errors [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2015: 630-656

[95] Mukherjee P, Wichs D. Two round multiparty computation via multi-key FHE [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2016: 735-763

[96] Peikert C, Shiehian S. Multi-key FHE from LWE, revisited [C] //Proc of Theory of Cryptography Conf. Berlin: Springer, 2016: 217-238

[97] Brakerski Z, Perlman R. Lattice-based fully dynamic multi-key FHE with short ciphertexts [C] //Proc of Annual Int Cryptology Conf. Berlin: Springer, 2016: 190-213

[98] Chen Long, Zhang Zhenfeng, Wang Xueqing. Batched multi-hop multi-key FHE from ring-LWE with compact ciphertext extension [C] //Proc of Theory of Cryptography Conf. Berlin: Springer, 2017: 597-627

[99] Chen Hao, Dai Wei, Kim M, et al. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference [C] //Proc of the 2019 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 395-412

[100] Chen Hao, Chillotti I, Song Y. Multi-key homomorphic encryption from TFHE [C] //Proc of Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2019: 446-472

[101] Lu Wenjie, Huang Zhicong, Chen Hong, et al. PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption [OL]. 2020 [2021-07-29]. https://eprint.iacr.org/2020/1606

[102] Wang Xiao, Malozemoff A J, Katz J. EMP-toolkit: Efficient MultiParty computation toolkit [J/OL]. 2016 [2021-07-29]. https://github.com/emp-toolkit

[103] Zahur S, Evans D. Obliv-C: Alanguage for extensible data-oblivious computation [OL]. 2015 [2021-07-29]. https://eprint.iacr.org/2015/1153.pdf

[104] Liu Chang, Wang Xiao Shaun, Nayak K, et al. Oblivm: A programming framework for secure computation [C] //Proc of the 2015 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2015: 359-376

[105] Franz M, Holzer A, Katzenbeisser S, et al. CBMC-GC: An ANSI C compiler for secure two-party computations [C] //Proc of Int Conf on Compiler Construction. Berlin: Springer, 2014: 244-249

[106] Mood B, Gupta D, Carter H, et al. Frigate: A validated, extensible, and efficient compiler and interpreter for secure computation [C] //Proc of the 2016 IEEE European Symp on Security and Privacy. Piscataway, NJ: IEEE, 2016: 112-127

[107] Keller M. MP-SPDZ: A versatile framework for multi-party computation [C] //Proc of the 2020 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 1575-1590

[108] Rastogi A, Hammer M A, Hicks M. Wysteria: A programming language for generic, mixed-mode multiparty computations [C] //Proc of the 2014 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2014: 655-670

[109] Zhang Yihua, Steele A, Blanton M. PICCO: A general-purpose compiler for private distributed computation [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 813-826

[110] Demmler D, Schneider T, Zohner M. ABY-A framework for efficient mixed-protocol secure two-party computation [C/OL] //Proc of ISOC Network and Distributed System Security Symp. Reston VA: The Internet Society, 2015 [2021-07-29]. https://encrypto. de/papers/DSZ15.pdf

[111] Mohassel P, Zhang Yupeng. Secureml: A system for scalable privacy-preserving machine learning [C] //Proc of the 2017 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2017: 19-38

[112] Gilad-Bachrach R, Dowlin N, Laine K, et al. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy [C] //Proc of Int Conf on Machine Learning. New York: ACM, 2016: 201-210

[113] Liu Jian, Juuti M, Lu Yao, et al. Oblivious neural network predictions via MiniONN transformations [C] //Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 619-631

[114] Riazi M S, Weinert C, Tkachenko O, et al. Chameleon: A hybrid secure computation framework for MAChine learning applications [C] //Proc of the 2018 on Asia Conf on Computer and Communications Security. New York: ACM, 2018: 707-721

[115] Juvekar C, Vaikuntanathan V, Chandrakasan A. GAZELLE: A low latency framework for secure neural network inference [C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 1651-1669

[116] Mishra P, Lehmkuhl R, Srinivasan A, et al. Delphi: A cryptographic inference service for neural networks [C] //Proc of the 29th USENIX Security Symp. Berkeley, CA: USENIX Association, 2020: 2505-2522

[117] Rouhani B D, Riazi M S, Koushanfar F. DeepSecure: Scalable provably-secure deep learning [C] //Proc of the 55th Annual Design Automation Conf. New York: ACM, 2018: 1-6

[118] Riazi M S, Samragh M, Chen H, et al. XONN: XNOR-based oblivious deep neural network inference [C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 1501-1518

[119] Agrawal N, Shahin Shamsabadi A, Kusner M J, et al. QUOTIENT: Two-party secure neural network training and prediction [C] //Proc of the 2019 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 1231-1247

[120] Kumar N, Rathee M, Chandran N, et al. CryptFlow: Secure tensorflow inference [C] //Proc of the 2020 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2020: 336-353

[121] Li Yi, Xu Wei. PrivPy: General and scalable privacy-preserving data mining [C] //Proc of the 25th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining. New York: ACM, 2019: 1299-1307

[122] Shokri R, Shmatikov V. Privacy-preserving deep learning [C] //Proc of the 22nd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2015: 1310-1321

[123] Aono Y, Hayashi T, Wang L, et al. Privacy-preserving deep learning via additively homomorphic encryption [J]. IEEE Transactions on Information Forensics and Security, 2017, 13(5): 1333-1345

[124] 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

Guo Juanjuan, born in 1996. PhD. Her main research intrersts include secure multiparty computation and applied cryptography.

郭娟娟,1996 年生.博士.主要研究方向为安全多方计算和应用密码学.

Wang Qiongxiao, born in 1982. PhD, senior engineer. Her main research interests include applied cryptography and identity management.

王琼霄,1982 年生.博士,高级工程师.主要研究方向为应用密码学和身份管理.

Wang Tianyu, born in 1989. Master, senior researcher. His main research interests include cryptography and data security.

王天雨,1989 年生.硕士,高级研究员.主要研究方向为密码学和数据安全.

Xu Xin, born in 1993. PhD. Her main research interests include distributed system and applied cryptography.

许 新,1993 年生.博士.主要研究方向为分布式系统和应用密码学.

Lin Jingqiang, born in 1978. PhD, professor, member of CCF. His main research interests include applied cryptography, network security and system security.

林璟锵,1978 年生.博士,教授,CCF 会员.主要研究方向为应用密码学、网络安全和系统安全.

《计算机研究与发展》征订启事

《计算机研究与发展》(Journal of Computer Research and Development)是中国科学院计算技术研究所和中国计算机学会联合主办、科学出版社出版的学术性刊物,中国计算机学会会刊.主要刊登计算机科学技术领域高水平的学术论文、最新科研成果和重大应用成果.读者对象为从事计算机研究与开发的研究人员、工程技术人员、各大专院校计算机相关专业的师生以及高新企业研发人员等.

《计算机研究与发展》于 1958 年创刊,是我国第一个计算机刊物,现已成为我国计算机领域权威性的学术期刊之一.并历次被评为我国计算机类核心期刊,多次被评为“中国百种杰出学术期刊”.此外,还被《中国学术期刊文摘》、《中国科学引文索引》、“中国科学引文数据库”、“中国科技论文统计源数据库”、美国工程索引(Ei)检索系统、日本《科学技术文献速报》、俄罗斯《文摘杂志》、英国《科学文摘》(SA)等国内外重要检索机构收录.

国内邮发代号:2-654;国外发行代号:M603

国内统一连续出版物号:CN11-1777/TP

国际标准连续出版物号:ISSN1000-1239

联系方式:

100190 北京中关村科学院南路 6 号《计算机研究与发展》编辑部

电话: +86(10)62620696(兼传真);+86(10)62600350

Email:crad@ict.ac.cn

https://crad.ict.ac.cn