GM/Y 5003-2024

多方安全计算密码技术研究

密码行业标准化技术委员会
CRYPTOGRAPHY STANDARDIZATION TECHNICAL COMMITTEE

2024 年 12 月

摘要

随着云计算、大数据、人工智能等信息技术的不断发展，用户数据脱离自身可控环境、海量用户数据集中处理等数据处理方式成为常态，在提供数据服务、挖掘数据价值的过程中需要解决数据隐私保护的问题。多方安全计算（Secure Multiparty Computation，MPC）是为数据安全融合而生的安全技术，可以在保护个人隐私的前提下进行协同计算，解决数据使用中“可用性”和“隐私性”之间的矛盾。本文归纳了 MPC 协议的系统功能、安全模型、安全目标、通信模型等基本要素与特征，对四种典型 MPC 技术路线（混淆电路、不经意传输、秘密分享、同态加密）的基本原理、典型协议实现、协议的性能优化和安全性增强等方面展开了详细的介绍。经过四十多年的研究，MPC 已有较为成熟的技术方案，有着广阔的应用前景，涵盖深度学习、隐私集合求交、隐私查询等多种领域，出现了大量的应用系统和开源软件工具。本文还对 MPC 相关标准现状和知识产权情况进行了分析。

关键词：多方安全计算 混淆电路 不经意传输 秘密分享 同态加密

目录

前言

多方安全计算经过四十多年的发展，理论已非常成熟。从 2018 年起国内外 MPC 产业开始快速成长，国际上相继成立了 MPC 产业联盟、MPC 标准组织等，我国在通信、金融等行业也推出了 MPC 应用标准，但是目前仍然缺乏对 MPC 基础密码技术和协议的规范。本技术报告通过梳理国内外 MPC 理论成果、产业应用和标准现状，明确 MPC 概念范畴、技术框架和安全要求。

本研究报告是由密码行业标准化技术委员会根据国家密码管理局批准下达的密码行业标准研究任务。项目名称为《多方安全计算密码技术研究》，项目类型为标准研究类项目，项目所属工作组为基础工作组。

项目承担单位：中国科学技术大学。

项目参与单位：中国科学院信息工程研究所、北京中科研究院、中国科学院数据与通信保护研究教育中心、华控清交信息科技（北京）有限公司、北京航空航天大学、山东大学、联想（北京）有限公司、智慧足迹数据科技有限公司、三未信安科技股份有限公司。

项目参与人员：林璟锵、徐葳、王琼霄、郭娟娟、李艺、王云河、王伟、李冰雨、伍前红、王天雨、孔凡玉、高志权、黄卓磊、成佳、张岩。

多方安全计算密码技术研究

1. 概述

近年来大数据、人工智能、云计算等新兴技术不断发展，引领数字经济发展浪潮，在金融、政务、电商等多个行业领域发挥着巨大作用。2016 年 12 月，国务院印发“十三五”国家战略性新兴产业发展规划，明确要加快建设“数字中国”，推动物联网、云计算和人工智能等技术向各行业全面融合渗透，构建万物互联、融合创新、智能协同、安全可控的新一代信息技术产业体系。数据在新兴技术和产业应用中发挥着巨大的作用。党的十九届四中全会首次将数据列为重要生产要素。2020 年以来，中共中央、国务院陆续发布《关于构建更加完善的要素市场化配置体制机制的意见》、《关于新时代加快完善社会主义市场经济体制的意见》等，提出加快培育数据要素市场的具体意见，对建立数据生产要素流动机制、提高数据要素质量和配置效率具有重大意义。

多元数据的汇聚、融合、挖掘将进一步释放数据潜能，发挥数据的巨大应用价值。数据的使用涉及多源化的数据来源，存在泄露信息的安全风险，一方面数据被企业作为重要的商业秘密和私有资产，数据融合导致的泄密风险会给企业带来巨大经济利益损失，另一方面数据融合过程中可能泄露个体隐私信息，使企业等实体组织面临相关法规的处罚或者名誉损失。国家互联网信息办公室 2019 年 5 月就《数据安全管理办法（征求意见稿）》向社会公开征求意见，其它包括《数据安全法》、《网络安全法》、《个人信息保护法》、《关于银行业金融机构做好个人金融信息保护工作的通知》等行政法规和规章制度也对隐私保护做了相关规定。其中，《数据安全法》明确了数据安全主管机构的监管职责，建立健全数据安全协同治理体系，提高数据安全保障能力，促进数据开发利用，为数字化经济的安全健康发展提供了有力支撑。

密码技术是解决数据融合应用问题的重要途径。《中华人民共和国密码法》已于 2020 年 1 月 1 日起施行，密码技术将在国家重要行业领域发挥越来越重要的作用。因此，通过使用密码技术实现多个实体组织间的可信互联、安全互通，解决数据融合发展过程中的信息泄漏问题，将有利于增强企业合理合规使用数据的信心，有利于促进新兴技术的发展。

多方安全计算（Secure Multiparty Computation，MPC）作为密码学领域的一个重要分支，是专门为数据安全融合而生的技术。MPC 理论是姚期智院士于 1982 年通过提出和解答著名的百万富翁问题而创立，其基本原理是在没有可信第三方的前提下，两个百万富翁如何在不泄露自己财产的情况下来比较谁更富有。MPC 中多个参与方使用各自的秘密输入协同计算某个函数，即使在一方甚至多方被攻击的情况下，仍能保证参与方的原始秘密数据不被意外泄露，并且保证函数计算结果的正确性和隐私安全，从而实现了数据的“可用而不可见”。

MPC 与传统数据加密技术存在不同，支持数据以非明文的方式参与计算。传统数据加密技术通过对数据加密、物理防护等措施实现数据在存储、传输等环节的安全保护，结合身份鉴别、访问控制等技术，确保数据明文仅能被授权访问者获取。基于传统的数据加密技术，数据在使用环节仍需以明文的方式出现并参与运算，尤其在多方计算的环境中，明文参与计算的方式无法提供对不同参与方的敏感数据安全保护。MPC 可以支持数据以非明文的方式参与计算，因此多个参与

方可以让自己的数据参与共同的计算任务而又保证各自的数据明文不被其他参与方获取。

自 MPC 理论创立以来，相关理论和技术分支得到了长足发展，MPC 技术实现层面也取得了很多进展。针对 MPC 技术的可用性、安全性的问题，国内外学者长期以来开展了深入的研究，基于不同的技术原理提出了一系列 MPC 协议，并提出了一些具有实用性的开源系统框架，MPC 技术已经达到了能够工程应用的成熟水平。与此同时，国内外出现了很多 MPC 技术初创公司，实现了通用 MPC 计算能力或专用功能的 MPC 服务平台/产品，在金融、政务等行业领域已经完成了落地应用。目前 MPC 技术已处于大规模应用的初始阶段。

MPC 技术应用和产业发展的同时，MPC 相关标准化工作也在推进，但密码技术相关标准仍匮乏。国内金融、通信领域已经推出 MPC 应用规范和检测标准。然而，MPC 作为重要的密码技术分支，仍缺少核心技术标准及与密码应用相关的技术标准，MPC 技术使用和 MPC 产品推广都缺乏密码合规性依据。

本报告通过对 MPC 理论发展现状、技术应用现状、标准现状进行系统全面的梳理，探索 MPC 一般技术框架，为 MPC 行业应用提供技术参考。

2. 基础技术原理

MPC 是指在一个分布式的网络中，每个参与方都各自持有秘密输入，希望共同完成对某个函数的计算，但是要求每个参与方除计算结果外均不能得到其他参与实体的任何输入信息。

自 MPC 理论创立以来，在四十多年的时间里多个理论分支都得到了长足发展。常用的基础技术包括混淆电路（Garbled Circuit,GC）、不经意传输（Oblivious Transfer,OT）、秘密分享（Secret Sharing,SS）和同态加密（Homomorphic Encryption,HE）等。

2.1 混淆电路

混淆电路作为 MPC 核心技术之一被广泛研究，它是将两方参与的安全计算函数编译成布尔电路的形式，并将电路真值表加密打乱，从而实现电路的正常输出但又不泄露参与计算的双方的私有数据。

混淆电路的参与方包括混淆器 Garbler 和评估器 Evaluator。Garbler 根据算法逻辑生成电路，包括若干个基本电路门 XOR、AND 等，每个基本电路门通常有两个输入导线和一个输出导线，Garbler 为每个导线值为 0 和 1 选取对应的随机数作为标签（也称为密钥），并使用输入导线标签组合作为密钥加密输出导线标签，得到基本电路门的混淆表。Evaluator 使用输入导线标签，解密混淆电路得到输出导线标签，并用输出标签解密输出门（最终输出的电路门）获得真正的输出值。

计算基本电路门时，Garbler 和 Evaluator 分别持有输入比特v_a和v_b。计算步骤分为三步：Garbler 生成混淆表，Garbler 和 Evaluator 之间传输标签和混淆表，Evaluator 评估电路，如图 1 所示：

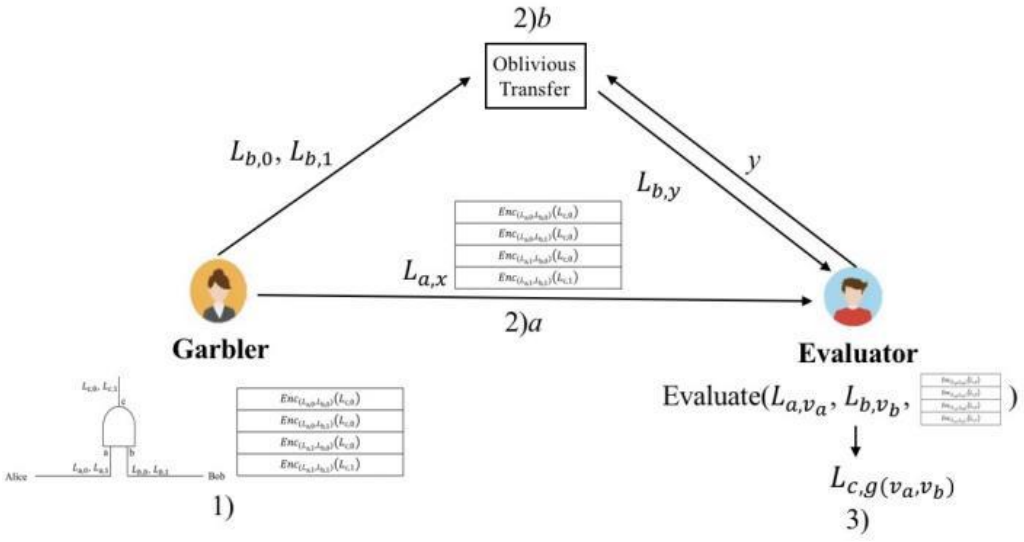

图 1　GC 基本流程

1)　Garbler 生成混淆表

以 AND 门为例，Garbler 为每个导线取值为 0 和 1 分别选取对应的标签，记作$L_{i,0}$、$L_{i,1}$（下标i表示导线索引，下标 0、1 代表导线真实取值），如图 2 所示。

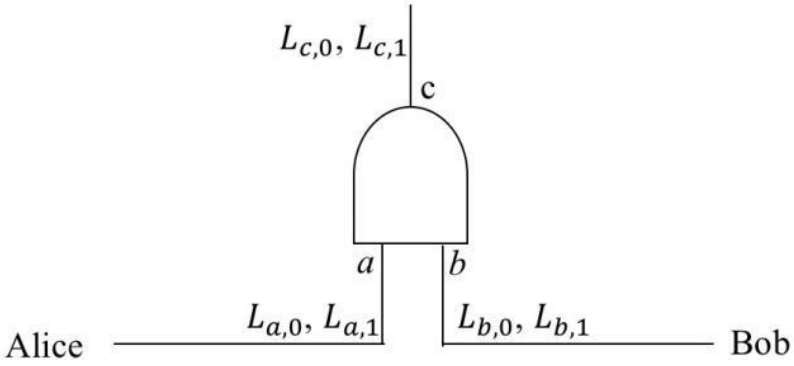

图 2　GC 基本电路门

对于每个电路门，Garbler 会根据电路门的真值表，使用输入导线标签对输出导线标签进行加密，得到一个四行的混淆表。输入导线索引为a、b，对应的取值分别为v_a、v_b（分别由 Garbler 和 Evaluator 持有）；输出导线索引为c，取值为v_c。以“AND”门为例，AND 门真值表及其对应的标签如表 1 所示。

表 1　AND 门真值表及其对应的标签

AND 门真值表			AND 门真值表对应的标签		
输入值v_a	输入值v_b	输出值v_c	输入值v_a	输入值v_b	输出值v_c
0	0	0	$L_{a,0}$	$L_{b,0}$	$L_{c,0}$
0	1	0	$L_{a,0}$	$L_{b,1}$	$L_{c,0}$
1	0	0	$L_{a,1}$	$L_{b,0}$	$L_{c,0}$
1	1	1	$L_{a,1}$	$L_{b,1}$	$L_{c,1}$

对真值表进行加密时，每一行使用输入导线a、b的真值对应的标签作为密钥，加密输出导线c的真值对应的标签。AND 门的真值表加密后如表 2 所示的加密表。最后，Garbler 打乱四行密文的排列顺序得到混淆表。

表 2　加密表

混淆表
$Enc_{(L_{a,0},L_{b,0})}(L_{c,0})$
$Enc_{(L_{a,0},L_{b,1})}(L_{c,0})$
$Enc_{(L_{a,1},L_{b,0})}(L_{c,0})$
$Enc_{(L_{a,1},L_{b,1})}(L_{c,1})$

当电路门是 XOR、OR 等门时，采用和 AND 门相同的方法，用输入导线的标签加密输出导线的标签得到加密表，打乱密文排列得到最终的混淆表。

2) Garbler 和 Evaluator 之间传输数据

a) Garbler 向 Evaluator 发送混淆表，以及 Garbler 的输入对应的标签L_{a,v_a}。

b) Garbler 和 Evaluator 之间执行不经意传输：Garbler 的输入是 Evaluator 输入导线对应的两个标签$L_{b,0}$、$L_{b,1}$，Evaluator 的输入是导线b的取值v_b，OT 向 Evaluator 输出v_b对应的标签L_{b,v_b}。

3) Evaluator 评估电路

Evaluator 此时拥有自己的标签L_{b,v_b}、Garbler 的标签L_{a,v_a}以及混淆表。Evaluator 尝试用两个标签解密混淆表中的密文，获得输出标签L_{c,v_c}。当对电路逐层计算直到要评估输出门时，Evaluator 获得输出导线的标签后用它解密输出表，得到输出的真实值，输出$f(v_a,v_b)$。

在协议执行过程中，Garbler 和 Evaluator 只能获得最终电路的结果，无法得知对方的输入。对于 Garbler，OT 的安全性保证 Garbler 无法知道 Evaluator 的真实输入值，协议最后 Evaluator 向 Garbler 公布电路的最终输出。对于 Evaluator，获得 Garbler 的标签，通过 OT 获得自己输入对应的标签，在此过程中，无法得到 Garbler 的真实输入值。

2.2 不经意传输

不经意传输最早在 1981 年被 Michael O.Rabin 提出[1]，是一个重要的密码学基础工具，可以被应用到许多密码学协议的构造中，例如混淆电路、隐私信息检索等。在不经意传输协议中有两个参与方，即发送方 S 和接收方 R，发送方有 n 个输入值，接收方希望得到其中某一个值，与此同时协议要保证发送方不知道接收方的选择信息，且接收方除了得到自己选择的值以外不知道其它输入值的相关信息。不经意传输协议分为 2 选 1、n 选 1、n 选 k 等，其中，2 选 1 不经意传输可以记作 1-out-of-2 OT。

1-out-of-2 OT 的原理如图 3 所示：OT 的输入是 S 创建的两个消息以及 R 提供的比特b，输出是给 R 的消息m_b。R 在不向 S 透露比特b的情况下接收m_b，S 确保 R 只能获得m_b而不能知道m_{1-b}。

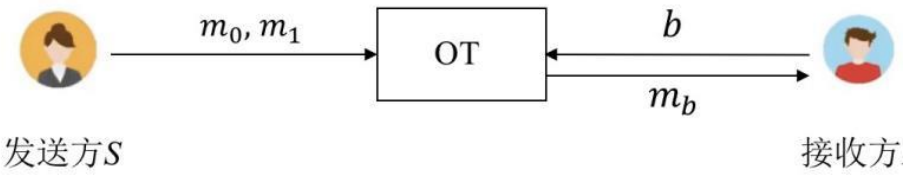

图 3　1-out-of-2 OT

OT 在最初提出的时候使用 RSA 算法来实现：

1）　S 生成 RSA 密钥对，包括模数n，公共指数e和私有指数d。生成两个随机数x_0、x_1和n、e一起发送给 R。

2）　R 选择一个随机值k，计算$v = (x_b + k^e)\ mod\ n$，将v发给 S。

3）　S 计算：$k_0 = (v - x_0)^d \bmod n$，$k_1 = (v - x_1)^d\ mod\ n$。使用$k_0$、$k_1$来加密两个秘密$m_0$、$m_1$：$m_0' = m_0 + k_0$，$m_1' = m_1 + k_1$，将$m_0'$、$m_1'$发给 R。

4）　R 使用k可以解开两个消息中的一条，解密得到$m_b = m_b' - k$。

使用 OT 协议进行多方安全计算，可以计算任意的比特运算函数[3]。所有的比特运算都可以分解为 XOR、AND 及 NOT 运算的组合，计算过程分为三个步骤：

输入阶段：n个参与方$P_1, \dots, P_n$拥有各自的函数输入自变量$x_i (i = 1,2, \dots, n) \in \{0,1\}$，$P_i$将其$x_i$随机分解为：$x_i = x_{i,1} \oplus x_{i,2} \oplus \dots \oplus x_{i,n}$，并将$x_{i,j}$发送给$P_j$。

计算阶段：

对于 XOR 和 NOT 运算，参与方可以在本地对份额进行计算，无需使用 OT。AND 运算过程中需要调用 OT，下面主要介绍 AND 运算。

AND 运算的输入分别是a、b（可以是初始输入，也可以是计算的中间结果），且a、b被分解成$a = a_1 \oplus a_2 \oplus \dots \oplus a_n$，$b = b_1 \oplus b_2 \oplus \dots \oplus b_n$，其中$P_i$拥有$a_i$、$b_i$。AND 运算的目标是使得$P_i$经过计算获得$c_i$，且满足$c_1 \oplus c_2 \oplus \dots \oplus c_n = a \cdot b = (a_1 \oplus a_2 \oplus \dots \oplus a_n) \cdot (b_1 \oplus b_2 \oplus \dots \oplus b_n)$。

当$n = 2$时，AND 运算的目标是使P_1得到c_1、P_2得到c_2，且 $c_1 \oplus c_2 = (a_1 \oplus a_2) \cdot (b_1 \oplus b_2)$。令$P_1$作为OT的发送方，$P_2$作为OT的接收方，$P_1$随机地选择$c_1 \in \{0,1\}$，并设置参数$d_1 = c_1 \oplus (a_1 \cdot b_1)$、$d_2 = c_1 \oplus [a_1 \cdot (b_1 \oplus 1)]$、$d_3 = c_1 \oplus [(a_1 \oplus 1) \cdot b_1]$、$d_4 = c_1 \oplus [(a_1 \oplus 1) \cdot (b_1 \oplus 1)]$为 OT 的输入。$P_2$设置$l = 1 + 2a_2 + b_2 \in \{1,2,3,4\}$作为 OT 的输入。执行 OT 协议后，$P_2$只能得到$d_l = d_{1+2a_2+b_2}$，$P_2$设置$c_2 = d_l$。容易验证$c_1 \oplus c_2 = (a_1 \oplus a_2) \cdot (b_1 \oplus b_2)$，例如$P_1$拥有$(a_1, b_1) = (1,0)$、$P_2$拥有$(a_2, b_2) = (0,1)$，$P_2$计算$l = 1 + 2a_2 + b_2 = 2$，因此设置$c_2 = d_2 = c_1 \oplus [a_1 \cdot (b_1 \oplus 1)]$。此时$c_1 \oplus c_2 = c_1 \oplus c_1 \oplus [a_1 \cdot (b_1 \oplus 1)] = a_1 \cdot (b_1 \oplus 1) = 1$，因此$(a_1 \oplus a_2) \cdot (b_1 \oplus b_2) = (1 \oplus 0) \cdot (0 \oplus 1) = 1$，满足$c_1 \oplus c_2 = (a_1 \oplus a_2) \cdot (b_1 \oplus b_2)$。

对于一般情况，由于$(\oplus_{i=1}^n a_i) \cdot (\oplus_{i=1}^n b_i) = [\oplus_{i=1}^n (a_i \cdot b_i)] \oplus \{\oplus_{1 \le i < j \le n} [(a_i \cdot b_j) \oplus (a_j \cdot b_i)] = (n-2)[\oplus_{i=1}^n (a_i \cdot b_i)] \oplus \{\oplus_{1 \le i < j \le n} [(a_i \oplus a_j) \cdot (b_i \oplus b_j)]\}$，$P_i$可以在本地计算$(n-2)(a_i \cdot b_i) = \oplus_{i=1}^{n-2} (a_i \cdot b_i)$。任意$P_i$和$P_j$都可以使用 OT 来计算$(a_i \oplus a_j) \cdot (b_i \oplus b_j)$，OT 的使用与上文$n = 2$时相同。

输出阶段：假设经过最后一步运算后P_i得到y_i，P_i将y_i广播后所有参与方获得函数最终的输出结果$y = y_1 \oplus y_2 \oplus \dots \oplus y_n$。

2.3 秘密分享

秘密分享的概念最早由 Adi Shamir 和 G.R.Blakley 于 1979 年提出。秘密分享是将秘密拆分给多个参与方，每个参与方都持有秘密的一部分（称为秘密份额），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。秘密分享的参与方有：秘密分发方 Dealer、计算参与方$P_1, P_2, \ldots, P_n$以及秘密恢复方 Receiver（可以是份额持有方），如图 4 所示。

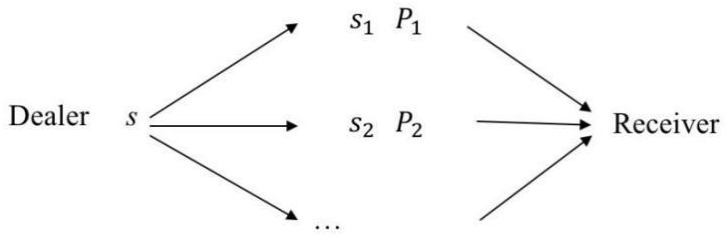

图 4　秘密分享

秘密分享主要包含以下几个阶段：秘密分发、函数计算、秘密恢复。秘密分发过程中，Dealer 为秘密s生成份额$s_i(i = 1, \ldots, n)$，将份额分发给对应的参与方P_i。函数计算过程中，份额持有方们对份额进行计算，得到函数结果的份额。秘密恢复过程中，Receiver 对函数结果的份额进行秘密恢复，得到函数计算结果。

根据秘密分享原理可以分为基于多项式的秘密分享和加性秘密分享。基于多项式的秘密分享的典型方案是 Shamir 门限秘密分享，加性秘密分享的典型方案是 Beaver 三元组。

1）Shamir 秘密分享

Shamir 门限秘密分享[4]是基于拉格朗日插值多项式构造的。Shamir(t,n)门限秘密分享中，原始秘密为s，存在n个秘密份额持有方，大于等于t个份额持有方可以共同恢复出秘密s。

a）秘密分发。Dealer 选取$\mathrm{t}-1$个正整数$a_1, \ldots, a_{t-1}$，令$a_0 = s$，$a_i < q$，构建多项式$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{t-1} x^{t-1}$。秘密分发方在多项式上选$n$个点，例如令$i = 1, \ldots, n$计算获得$(i, f(i))$，通过安全信道为每个参与方$P_i$分发$(i, f(i))$。

b）秘密恢复。在获得至少t个参与方的秘密份额的情况下，使用拉格朗日多项式插值法，可以恢复出秘密s。

Shamir 秘密分享支持加法和乘法运算，具体流程如下：

a）秘密分发。设秘密s_1的多项式为$f_1(x)$，秘密s_2的多项式为$f_2(x)$，份额持有方P_i拥有两个秘密的份额 $(i, f_1(i))$，$(i, f_2(i))$。

b）函数计算。

加法：P_i在本地将s_1和s_2份额相加得到$s_1 + s_2$的份额：$f_1(i) + f_2(i)$。

乘法：P_i在本地对份额相乘$f_1(i) \times f_2(i)$，并做随机化和降阶处理。BGW 是在有限域F_2上实现的基于 Shamir 秘密分享的方案，对乘法运算的随机化和降阶处理给出了详细的介绍，具体流程见 3.2.3.1 节。

c）计算结果秘密恢复。参与方将函数计算得到的结果作为秘密份额，至少t个参与方使用拉格朗日多项式插值法，可以重构多项式并恢复出计算结果$s_1 + s_2$或$s_1 \cdot s_2$。

2) Beaver 三元组

Beaver 三元组[5]由 Beaver 于 1991 年提出，主要用于 MPC 协议中的乘法计算，计算过程如下：

a) 预处理。生成 Beaver 三元组(a,b,c)，其中$c=ab$。计算a、b、c的加性秘密分享份额a_i、b_i、$c_i(1\le i\le n)$，满足$\sum_{i=1}^{n}a_i=a$、$\sum_{i=1}^{n}b_i=b$、$\sum_{i=1}^{n}c_i=c$。将三元组份额a_i、b_i、c_i分发给参与方P_i。

b) 秘密分发。输入数据秘密分发：将私有数据x和y按照参与方数量n进行加性秘密拆分：$x=x_1+\cdots+x_n$、$y=y_1+\cdots+y_n$，将份额x_i、y_i分发给P_i。

c) 函数计算。

加法：P_i在本地将份额相加得到x_i+y_i。

乘法：P_i持有三元组份额a_i、b_i、c_i和数据份额x_i、y_i，在本地计算x_i-a_i、y_i-b_i，然后将其广播；每个P_i接收到广播的份额后加和得到$(x-a)$、$(y-b)$，结合三元组份额计算得到z的份额$z_i=(x-a)b_i+a_i(y-b)+c_i$。

d) 秘密恢复。

加法：将n个参与方的份额x_i+y_i相加得到$x+y$。

乘法：将n个参与方的份额$z_1,\ldots,z_n$相加得到$(x-a)b+a(y-b)+ab$，再与$(x-a)(y-b)$相加后，得到最终输出结果$x\cdot y$。则$z=x\cdot y=(x-a+a)(y-b+b)=(x-a)(y-b)+(x-a)b+a(y-b)+ab$成立。

Beaver 三元组是消耗性的，每次乘法计算都会消耗一个 Beaver 三元组，通过预先计算的 Beaver 三元组，将通信量和计算量移到了预处理阶段。

2.4 同态加密

同态密码的概念最初是由 Rivest 等人于 1978 年提出，同态加密指的是对几个数据的加密结果进行运算后再解密，得到的结果与这些数据未加密时执行某一运算所得的结果一致。同态加密可以解决云计算上的数据安全问题，使得云服务器可以合理利用密态数据，同时又不知道用户的隐私数据。

同态加密分为三种类型：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域同态运算：常见加法同态（Additively homomorphic encryption, AHE）或乘法同态（Multiplicative homomorphic encryption）；浅同态能够支持密文域有限次数的加法和乘法同态运算；全同态能实现任意次的加法和乘法同态运算。

目前常用的或主流的 HE 是公钥密码算法，HE 算法是概率多项式时间算法组成的密码体制：$\mathrm{HE}=(\mathrm{KeyGen},\mathrm{Enc},\mathrm{Eval},\mathrm{Dec})$，定义每个算法如下：

$\mathrm{HE.KeyGen}(1^{\lambda})\to(pk,sk)$：给定一个安全参数$\lambda$，密钥生成算法输出公钥$pk$，私钥$sk$。

$\mathrm{HE.Enc}(pk,m)\to c$：给定公钥$pk$和消息$m$，加密算法输出密文$c$。

$\mathrm{HE.Eval}(pk,f,c,c')\to c_{eval}$：给定公钥$pk$，两个密文$c$、$c'$，函数$f$，评估算法输出评估密文$c_{eval}=f(\mathrm{c},c')$。

$\mathrm{HE.Dec(sk,c)} \rightarrow m$：给定密文$c$和私钥$sk$，解密算法输出明文$m$。

部分同态加密算法中，加法同态加密应用最广泛的为 Paillier 加密算法，乘法同态加密算法有 ElGamal 算法。

1） Paillier 加密系统

Paillier 加密系统[6]是 1999 年 Paillier 提出的公钥密码方案，基于复合剩余类的困难问题。Paillier 算法支持加法同态，以及密文与明文的乘法同态。

Paillier 算法的基本工作流程如下：

$\mathrm{Paillier.KeyGen}$：随机选择两个大素数p、q满足$\gcd\big(pq,(p-1)(q-1)\big)=1$。计算$n=p\cdot q$，$\lambda=lcm(p-1,q-1)$。选择随机整数$g\in Z_{n^2}^*$，使得满足$n$整除$g$的阶。公钥$pk$为$(n,g)$，私钥$sk$为 λ。

$\mathrm{Paillier.Enc}$：选取随机整数$r\in Z_{n^2}^*$，对于明文$m\in Z_n$，加密后的密文为$c=g^m\cdot r^n\ mod\ n^2$。

$\mathrm{Paillier.Dec}$：$m=\frac{L(c^\lambda\ mod\ n^2)}{L(g^\lambda\ mod\ n^2)}\ mod\ n$，其中$L(u)=\frac{u-1}{n}$。

Paillier 算法支持加法同态，对两个密文数据相乘的结果进行解密，得到的值为两个明文数据的相加结果。$c_1\cdot c_2=(g^{m_1}\cdot r_1^n)\cdot(g^{m_2}\cdot r_2^n)=g^{m_1+m_2}(r_1r_2)^n=$
$\mathrm{Paillier.Enc}(m_1+m_2)mod\ n^2$，$\mathrm{Paillier.Dec}(c_1\cdot c_2)=m_1+m_2\ mod\ n$。

Paillier 算法不支持全密文运算的乘法同态，仅能支持m_1的密文与明文m_2的标量乘法同态，相当于是密文进行m_2次加法。对密文进行指定明文的指数运算，解密后得到两个明文相乘的计算结果，有如下：

$\mathrm{Paillier.Dec(Paillier.Enc}(m_1)^{m_2})=m_1m_2\ mod\ n$。

2） ElGamal 算法

ElGamal 加密系统是 1985 年 ElGamal 提出的基于有限域上离散对数难题的公钥密码算法。ElGamal 算法支持乘法同态。

$\mathrm{ElGamal.KeyGen}$：选取随机大素数p，选定素域F_p的一个生成元g，产生随机数x，计算$y=g^x\ mod\ p$。公开(p,g,y)为公钥，x为私钥。

$\mathrm{ElGamal.Enc}$：生成随机数r，对于明文消息m计算密文$c=(c_1,c_2)$，$c_1=g^r\ mod\ p$，$c_2=my^r\ mod\ p$。

$\mathrm{ElGamal.Dec}$：$c_2(c_1{}^x)^{-1}=m(g^x)^r(g^{rx})^{-1}=m\ mod\ p$。

ElGamal 算法满足乘法同态：有$c_1\cdot c_2=(g^{r_1}\ mod\ p,m_1y^{r_1}\ mod\ p)\cdot(g^{r_2}\ mod\ p,m_2y^{r_2}\ mod\ p)=(g^{r_1+r_2}\ mod\ p,m_1m_2y^{r_1+r_2}\ mod\ p)$，明文乘法运算对应密文乘法运算，所以 ElGamal 满足乘法同态属性。

3. 多方安全计算技术框架及进展

3.1 多方安全计算

本小节对 MPC 系统功能、安全模型、安全目标、通信模型进行介绍。

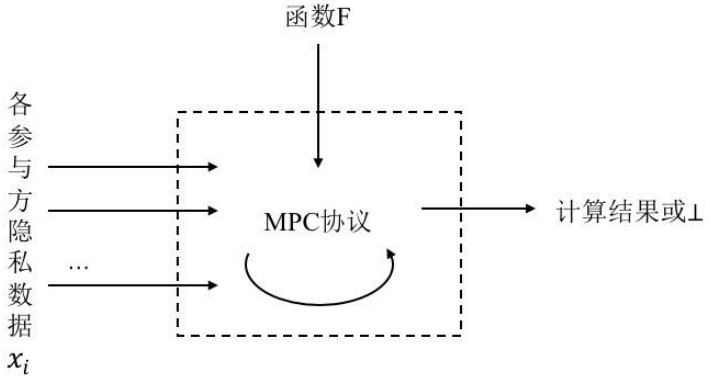

图 5　MPC 模型

MPC 是指多个参与方在不泄露各自隐私数据的前提下，共同完成某个函数计算的过程。如图 5 所示，各参与方P_i要基于各自的隐私数据x_i协同计算函数 F，各方需在 MPC 协议下执行操作，协议规定了每个参与方P_i在拥有安全参数、隐私输入、已接收信息等的情况下如何执行下一步操作：计算一个信息然后发送、或者停止输出（⊥）。

3.1.1 安全模型

MPC 协议的安全性总是在特定的敌手行为能力下进行考量，一般存在两种常用的敌手行为假设：

1) 半诚实（semi-honest）行为假设：系统中的敌手会按照协议规定进行操作，但会试图通过协议中得到的信息挖掘其它参与方的隐私数据。
2) 恶意（malicious）行为假设：攻击者不会遵守约定执行协议，会执行错误的协议操作、或者正确的协议操作，然后从得到的信息中推断其它参与方的隐私数据、甚至是以破坏协议执行为目标。恶意攻击者的行为是任意的、不可推理不可预测的。

根据系统对敌手的容忍程度，MPC 系统的安全模型可以分为：

1) 半诚实安全模型：该模型中保证用户隐私信息不被半诚实参与方获得。
2) 隐蔽（covert）安全模型：系统中的不诚实参与实体会试图改 变协议行为来挖掘其它参与实体的隐私，但是这种欺骗行为有一定的概率被其他参与方发现，隐蔽安全模型的安全性高于半诚实安全模型。公开可验证（public verifiable covert，PVC）在满足以上条件的基础上，进一步允许参与方可以生成公开可验证的欺骗证据。
3) 恶意安全模型：该模型中用户隐私数据不被恶意行为敌手获得，且不会因为敌手的恶意行为导致协议错误运行。

根据攻击者计算能力假设，MPC 系统可以分为计算安全模型和信息论安全模型。在计算安全模型中，攻击者的计算能力是概率多项式时间的，攻击者无法在有效时间内解决常见的困难问题。在信息论安全模型中，攻击者的计算能力是无限的。对于实际应用中的密码系统，只要达到计算安全性就可以满足实际的应用需求。

3.1.2 安全目标

MPC 协议一般围绕两个重要目标：

1) 参与方数据隐私安全。MPC 协议保证各参与方不应获得除其自身输出和可从其自身输入和输出中得出的信息以外的任何其他信息。
2) 计算结果正确性。MPC 协议保证最终的计算结果和基于原始数据明文的计算结果一致，或者保证发现敌手行为并停止计算。

进一步，对于参与方获得计算输出结果，MPC 协议能够达到不同程度的安全属性：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）。最强的安全属性是 GOD，无论攻击者的策略是什么，诚实参与方都能获得输出，即攻击者无法阻止诚实参与方获得输出；FN 的安全属性稍弱，只有诚实方能够收到输出的情况下，攻击者才能收到输出，即不允许出现攻击者获得输出但诚实参与方没有获得输出的情况；UA 安全属性弱于 FN，所有的诚实参与方都能获得输出，或者所有诚实参与方都不能获得输出；SA 安全属性最弱，攻击者可以阻止部分诚实参与方获得输出。

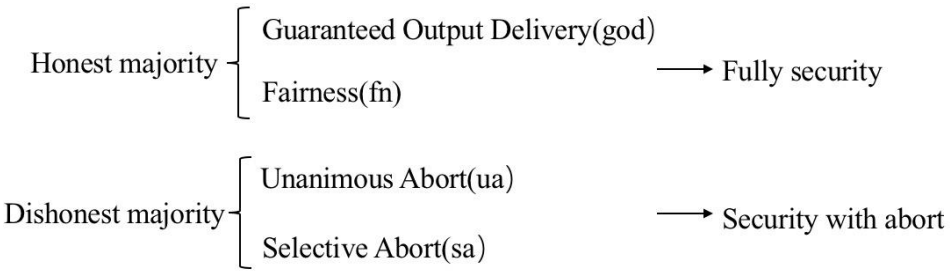

图 6　MPC 安全属性

攻击者能够通过各种手段控制部分参与方，参与方被控制后它的所有通信信息和本地信息都会被攻击者掌握。攻击者控制的参与方数量影响到协议的安全，（t,n）门限方案表示当参与方总数为 n 时，可容忍 t 个参与方被攻击者控制的 MPC 方案。当 t<(n/2)时诚实参与方占大多数（honest majority）；当 t≥(n/2)时不诚实方占大多数（dishonest majority），两方协议中有一个参与方损坏的情况也属于不诚实方占多数。

如图 6 所示，只有诚实占多数（honest majority）的情况下才能实现前两个属性，在不诚实占多数(dishonest majority)的情况下只能获得后两个属性。在两方方案中，存在一个恶意参与方（参与方总数的一半），也可以实现前两种属性。GOD 和 FN 又称为完全安全性（Fully security），UA 和 SA 又称为中止安全性（Security with abort）。

3.1.3 通信模型

MPC 方案的实施依赖于参与方之间的网络通信，其通信网络模型可以分为同步模型（Synchronous）和异步模型（Asynchronous）。

1) 同步通信模型：所有参与方共享一个全局时钟，在一个时钟周期开始时，保证所有参与方都能完成输入，然后执行本周期内的其他操作，计算并发出自己的消息。在同步模型中，消息传递时间的上限（Δ）是已知的，可以通过同步时钟和可靠信道来实现，参与方需要等到下一轮时钟才能继续下一步计算。同步模型保证了输入的完整性，可以容忍更多的不诚实参与方，但是计算速度慢，取决于网络最坏情况时延Δ。同步模型达

到完全安全性门限参与方总数至少为 n>2t，达到中止安全性参与方总数至少为 n>t。

2) 异步通信模型：不存在一个全局时钟，接收方收到的消息的时延Δ取决于实时的网络环境、事先不能预测。异步模型有较好的响应性，参与方可以至少收到 n-t 个输入，允许各方以实际网络的最快速度获得输出，但是具有较差的网络时延弹性、连接速度慢的参与方可能无法输入。异步模型参与方总数至少为 n>3t。

利用传统的密码技术可以对信道上传输的消息进行保护实现安全信道。常见安全信道有：点到点安全信道：保证通信双方消息的机密性，消息不会泄露给其他参与方；广播安全信道：一种是可信第三方对所有参与方进行广播，一种是参与方之间进行广播，广播信道要保证消息的机密性，每个参与方发送的消息及其身份标识符都可以被其它参与方收到。

3.2 技术方案

本节对 MPC 技术发展过程中的主要技术方案进行介绍。在第 2 章基础技术原理的基础上，本节介绍的方案进一步包含了对性能优化、安全性增强等方面的技术进展介绍。具体方案仍从混淆电路、不经意传输、秘密分享、同态加密等四个方面展开。

3.2.1 混淆电路

Yao 等人于 1982 年最早提出了多方安全计算的概念，于 1986 年给出了混淆电路的基础技术方案，称为 Yao's GC。混淆电路按照敌手模型可以分为半诚实行为假设和恶意行为假设，按照参与方数量分类可以分为两方计算和多方计算。

3.2.1.1 混淆电路性能优化

混淆电路的性能优化主要包括两个方面：一方面是通信效率的优化，指的是减少每个电路门所对应的密文的数量；另一方面是对计算开销的优化，指的是在生成和评估电路门时减少加解密次数以及改进加密方式。

表 3　混淆电路的性能优化（H 为杂凑算法）

技术	密文大小		每个门调用 H 的次数			
			Garbler		Evaluator	
	XOR	AND	XOR	AND	XOR	AND
Classical	4	4	4	4	4	4
Point-and-permute	4	4	4	4	1	1
Free-XOR	0	4	0	4	0	1
GRR+Free-XOR	0	3	0	4	0	1
GRR2	2	2	4	4	1	1
FleXOR	0,1,2	2	0,2,4	4	0,1,2	1
Half gates	0	2	0	4	0	2

表 3 总结了半诚实行为假设下二输入门的性能。最早的方案是 Yao's GC，之后提出的 Point-and-permute 对 Yao's GC 的计算开销进行优化，使得 Evaluator 只需解密一次密文就能获得输出导线标签。Free-XOR 方案是将 XOR

运算优化为只需要 0 个密文。GRR 与 Free-XOR 兼容，将 AND 门的密文优化为 3 个；而 GRR2[7]与 Free-XOR 不兼容，将每个门的密文数量减小为 2。FleXOR[8]将 Free-XOR 技术与 AND 门优化相结合，根据 XOR 门的输入是否是 AND 门的输出，XOR 门包含 0 到 2 个密文。Half gates 与 Free-XOR 兼容，将 AND 门的密文降到 2 个。

1) Yao's GC

对于二输入门的布尔电路，最早提出的混淆电路协议是 Yao's GC[2]。在电路生成阶段，Garbler 为每根导线选择两个标签，使用输入标签组合对输出标签进行加密得到密文，对真值表按顺序计算完密文后，然后对 4 个密文位置进行随机置换得到混淆表。在 Yao's GC 中，XOR 门和 AND 门开销也是一样的，每个门需要生成 6 个随机数，4 个密文，Garbler 需要计算四次两重对称加密，Evaluator 依次对四个密文解密直到解密成功为止，该方案通信开销和计算开销都很大需要进一步优化。

2) Point-and-permute

表 4 Point-and-permute 混淆表

密文	位置
$Enc_{(L_{a,0},L_{b,0})}(L_{c,0})$	(s_a^0, s_b^0)
$Enc_{(L_{a,0},L_{b,1})}(L_{c,0})$	(s_a^0, s_b^1)
$Enc_{(L_{a,1},L_{b,0})}(L_{c,0})$	(s_a^1, s_b^0)
$Enc_{(L_{a,1},L_{b,1})}(L_{c,1})$	(s_a^1, s_b^1)

Beaver 等人在 1990 年提出了 Point-and-permute[9]，Garbler 为电路门的输入导线i生成标签的同时，生成一个选择比特s_i，输入导线i取 0 和取 1 对应选择比特s_i^0、s_i^1取值相反，且与导线的真实值无关。此时输入导线a、b对应的标签为$(L_{a,0}, s_{a,0})$、$(L_{a,1}, s_{a,1})$、$(L_{b,0}, s_{b,0})$、$(L_{b,1}, s_{b,1})$，Garbler 根据导线的选择比特(s_a, s_b)来排列密文，如表 4 Point-and-permute 混淆表所示。例如，随机取$s_{a,0}=1$、$s_{b,0}=0$，那么$s_{a,1}=0$、$s_{b,1}=1$，密文$Enc_{(L_{a,0},L_{b,0})}(L_{c,0})$的位置为$(s_{a,0}, s_{b,0})=(1,0)$，应该是混淆表中的第 3 行。Evaluator 在解密时仅需解密选择比特位置对应的一个密文，减小了解密混淆表的计算开销。

3) Row reduction

Naor 等人在 1999 年提出 GRR3（gate row reduction）[10]，将每个门的混淆表密文数量从 4 行减少到了 3 行。将混淆表的第一行规定为 0，即$Enc_{(L_{a,0},L_{b,0})}(L_{c,0})=0$，Garbler 向 Evaluator 传输混淆表时，每个电路门只需要传输 3 个密文。如果 Evaluator 需要解密第一行密文，即对密文 0 进行解密获得输出导线标签$L_{c,0}=Dec_{L_{a,0},L_{b,0}}(0)$，这种方法对任意电路门都适用。令$L_{c,1}=L_{c,0}\oplus R$则可与 Free-XOR 兼容。2009 年 Pinkas 等人提出的 GRR2 使用多项式插值的性质，进一步将混淆表密文数量降为两行，但是该方法不能与 Free-XOR 兼容。

4) Free-XOR

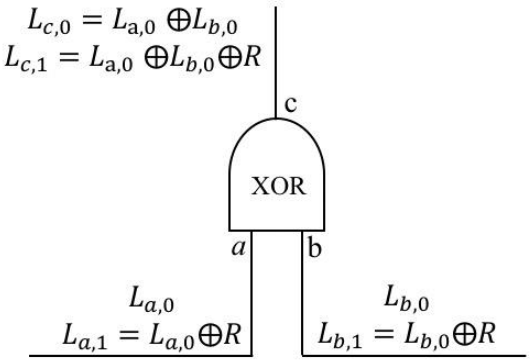

图 7　Free-XOR

Kolesnikov 等人在 2008 年提出 Free-XOR 技术[11]，是现有方案中计算 XOR 门的最优方案。在计算 XOR 门时，该方案不需要 Garbler 加密和传输混淆表，Evaluator 在本地即可计算 XOR 门输出导线的标签。如图 7 所示，对于每个导线i，取全局随机数R，Garbler 为导线i取 0 和取 1 分别生成标签$L_{i,0}$和$L_{i,1}$，满足关系$L_{i,1} = L_{i,0} \oplus R$。XOR 门输入线$a$、$b$的真实值分别为$v_a$、$v_b$，输出线$c$的真实值为$v_c$，Evaluator 从 Garbler 处获得输入导线标签L_{a,v_a}、L_{b,v_b}后可以直接计算得到输出标签$L_{c,v_c} = L_{a,v_a} \oplus L_{b,v_b}$。对于整个电路的输出门，用输出导线标签解密输出表中的密文，即可获得输出的明文值。

5)　Half gates

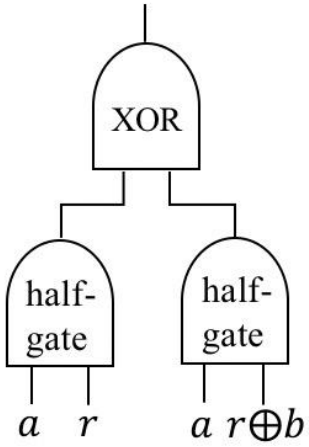

图 8　Half gates

Zahur 等人在 2015 年提出了半门（Half gates）[12]，该方案是现有方案中计算 AND 门的最优方案。在半门提出之前，计算 XOR 门的最优方案是 Free-XOR，通信开销是 0 个密文，与 Free-XOR 兼容的 AND 门的最优通信开销是 3 个密文。半门方案是与 Free-XOR 兼容的 AND 门优化，将 AND 门的计算转换为两个半门的异或，将 AND 门的通信开销降低到 2 个密文。该方案用到了 Free-XOR、GRR 和 Point-and-permute 等优化。方案设计思路如下：

Garbler-半门：$c = a \wedge b$，Garbler 知道a的值，Garbler 的输入为b，Garbler 为“Garbler-半门”生成两个密文：$H(L_{b,0}) \oplus L_{c,0}$，$H(L_{b,0} \oplus R) \oplus L_{c,0} \oplus aR$。如果$a = 0$，无论$b = 0$或$b = 1$，$c$取值都为 0，所以$b$的标签解密以上两个密文均获得$c = 0$对应的标签$L_{c,0}$。如果$a = 1$，若 $b = 0$，那么$c = 0$，所以用$L_{b,0}$解密第一

个密文得到标签$L_{c,0}$；若$b=1$，那么$c=1$，所以用$L_{b,0}\oplus R$解密第二个密文得到$c=1$对应的标签$L_{c,0}\oplus R$。

Evaluator-半门：$c=a\wedge b$，Evaluator 拥有输入a。Garbler 为“Evaluator-半门”生成 2 个密文：$H(L_{a,0})\oplus L_{c,0}$，$H(L_{a,0}\oplus R)\oplus L_{c,0}\oplus L_{b,0}$，用 Evaluator 的标签解密密文。如果$a=0$，那么$c=0$，用标签$L_{a,0}$解密获得输出标签$L_{c,0}$。如果$a=1$，用标签$L_{a,0}\oplus$R解密获得中间值$L_{c,0}\oplus L_{b,0}$，将中间值与导线$b$的标签异或：若$b=0$，则$c=0$，用标签$L_{b,0}$异或中间值得到输出标签$L_{c,0}$；若$b=1$，则$c=1$，用$L_{b,0}\oplus R$异或中间值得到输出标签$L_{c,0}\oplus R$。

Garbler-半门与 Evaluator-半门可以通过异或组成一个完整的与门，如图 8 所示。$c=a\wedge b=a\wedge(r\oplus r\oplus b)=(a\wedge r)\oplus(a\wedge(r\oplus b))$，Garbler 令 $r=\lambda_b$。其中导线i真实值是v_i，对应的选择比特是$s_i^{v_i}$，选择比特与真实值的关系满足$s_i^{v_i}=v_i\oplus\lambda_b$（$\lambda_b$称为掩码比特 mask bit 或混淆比特 permute bit）。因此 Garbler-半门是$a\wedge\lambda_b$，Garbler 的输入值是v_a；Evaluator-半门是$a\wedge(\lambda_b\oplus b)$，Evaluator 的输入是$\lambda_b\oplus v_b$。

Garbler 为 Garbler-半门、Evaluator-半门生成密文。对于 Garbler-半门，计算$f_G(v_a,\lambda_b)=v_a\wedge\lambda_b$，在 GRR 和 Point-and-permute 之前，两个密文为$H(L_{a,0})\oplus L_{GC,0}$，$H(L_{a,1})\oplus L_{GC,0}\oplus\lambda_b R$。在 GRR 和 Point-and-permute 之后，密文为$T_{GC}=H(L_{a,0})\oplus\ H(L_{a,1})\oplus\lambda_b R$。对于 Evaluator-半门，计算$f_E=v_a\wedge(v_b\oplus\lambda_b)$，在 GRR 之前，两个密文为$H(L_{b,\lambda_b})\oplus L_{EC,0}$，$H(L_{b,\lambda_b\oplus 1})\oplus L_{EC,0}\oplus L_{a,0}$。在 GRR 之后（不需要 permute），密文为$T_{EC}=H(L_{b,0})\oplus H(L_{b,1})\oplus L_{a,0}$。Garbler 发送 Evaluator 标签：$(L_{b,\lambda_b},0)$、$(L_{b,\lambda_b\oplus 1},1)$，Evaluator 将自己的真实输入$v_b$作为选择比特从两个标签中选择一个，如果$v_b=0$，选择标签$L_{b,\lambda_b}$，如果$v_b=1$，选择标签$L_{b,\lambda_b\oplus 1}$。然后，Garbler 发送自己的输入对应的标签$(L_{a,v_a},v_a\oplus\lambda_a)$。最后，Evaluator 用 Garbler 标签和 Evaluator 标签解密两个半门的密文，获得输出标签L_{GC}、L_{EC}，将L_{GC}和L_{EC}异或得到与门的输出标签。

6) GESS

GESS（Gate Evaluation Secret Sharing）[14]是由 Kolesnikov 于 2005 年提出的支持两个参与方进行安全计算的协议，是信息论安全的 GC 方案，其计算过程如图 9 所示：

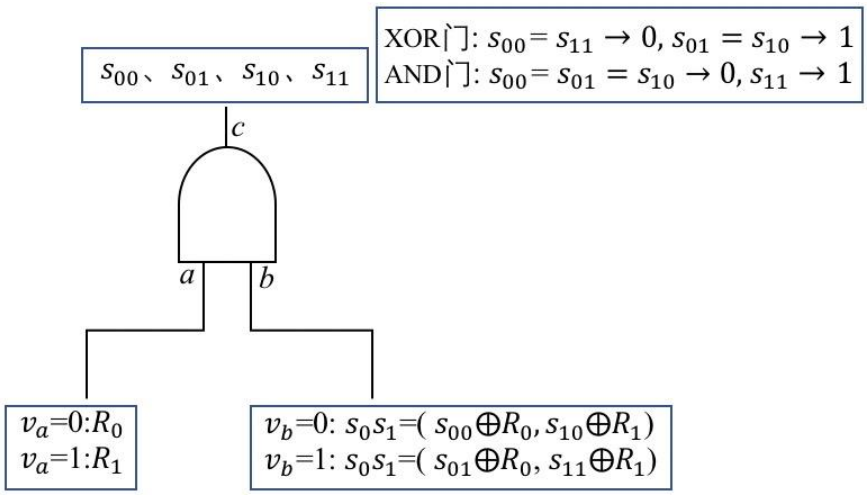

图 9 GESS

a) Garbler 首先确定输出标签：s_{00}、s_{01}、s_{10}、s_{11}作为四种输入所对应输出的标签，下标代表输入值。将输出标签与输出比特建立对应关系，以 XOR 运算为例，s_{00}=s_{11}对应于输出 0，s_{01}=s_{10}对应于输出 1；

b) Garbler 将输出标签进行秘密分享，得到两条输入线的输入 0 和 1 分别所对应的标签。Garbler 为输入导线a选择两个随机字符串R_0、R_1作为导线a取 0 和取 1 的标签，在R_0后面附加指针比特$p=0$以表示秘密恢复时需导线b的标签的左块s_0，在R_1后面附加指针比特$p=1$以表示秘密恢复时需导线b的标签的右块s_1。导线b取 0 时，标签为两个块s_0s_1：$s_0=s_{00}\oplus R_0$、$s_1=s_{10}\oplus R_1$；导线b取 1 时标签为两个块s_0s_1：$s_0=s_{01}\oplus R_0$、$s_1=s_{11}\oplus R_1$。按照以上方法 Evaluator 可以根据指针比特获知 Garbler 的输入比特。为了防止 Garbler 的隐私信息泄露，Garbler 随机选择指针比特p：如果取$p=0$，则按照上文的方法来分配份额;如果取$p=1$,Garbler 反转导线b标签中块的顺序,并反转导线a的附加指针位。

c) Evaluator 作为接收方,Garbler 作为发送方执行 OT 协议得到 Evaluator 真实输入对应的输入标签（导线b）。并且，Garbler 将其真实输入对应的输入标签（导线a）发送给 Evaluator。

d) Evaluator 使用自己的输入标签和 Garbler 的输入标签进行秘密恢复得到输出对应的标签，基于输出标签得到计算的输出。

上述构造方式效率较低,主要是因为导线b的份额大小是秘密值s大小的二倍，根据图 9 可得，导线b的两个份额有可优化的部分：当计算与门时导线b两个份额的左块相等，当计算或门时导线b两个份额的右块相等。使用这个性质可以避免相同数据的冗余，降低数据量。

7) Arithmetic-GC

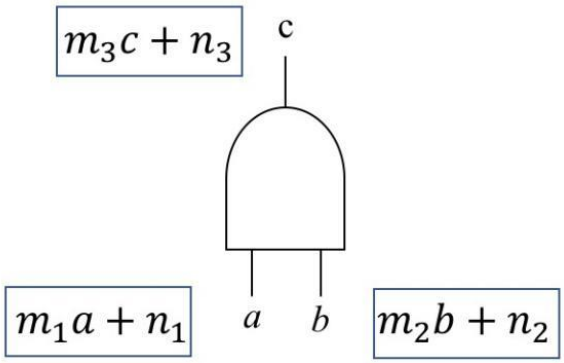

图 10 Arithmetic-GC

以上介绍的混淆电路都是计算布尔运算,也存在可以进行算术运算的混淆电路[13][15]，如图 10 所示，将输入/输出与标签之间的关系抽象为一般映射，采用表达式来描述标签与输入/输出数据之间的关系：$Q=ms+n$，其中s是输入/输出数据，m、n是随机数，Q是输入/输出数据对应的标签。该表达式也能兼容布尔运算。对于计算任务对应的算式，从数据输入开始逐层编码，最后基于输入标签得到输出标签。

3.2.1.2 混淆电路安全性增强

原始的 Yao's GC 只能抵抗半诚实行为敌手，然而实际应用中会存在恶意的敌手想要获得对方的输入或者导致协议计算得到错误的结果。将 Cut-and-Choose 技术应用到多方安全计算协议中，需要克服输入一致性问题和选择失败问题。输入一致性问题指的是，使用 Cut-and-Choose 时需要复制多份混淆电路，如果恶意参与方输入不一致则会影响协议的结果。选择失败攻击（selective failure attack）是由 OT 协议本身的缺陷带来的：在数据输入阶段双方执行 OT 时，对于 Evaluator 拥有的导线对应的 2 个标签值，Garbler 输入 1 个正确的标签和 1 个错误的标签，例如 Garbler 在 OT 中输入正确的L_0和错误的L_1，如果 Evaluator 的输入值为 0 它将会获得正确的标签并且继续电路的评估，如果 Evaluator 的输入值为 1 它将会退出协议，通过观察 Evaluator 的行为，Garbler 可以知道 Evaluator 的输入比特。2015 年蒋瀚等人对一致性问题和选择失败问题的现有解决方案进行了总结[19]。

混淆电路的安全性增强方案是指在恶意行为假设下仍具有安全性的方案，主要包括以下几种：Cut-and-Choose、三方 GC（3-GC）、PVC 安全 GC（PVC-GC）、Authentication GC（Auth-GC）和 n 方 GC（n-GC）。

1) Cut-and-Choose

Cut-and-Choose 技术在恶意安全模型的混淆电路中有很重要的作用。Pinkas 首次将 Cut-and-Choose 技术引入基于 Yao's GC 的安全两方计算协议[16]，基本思想是 Garbler 生成并发送s份（s是复制因子）混淆电路；Evaluator 随机检查其中的部分电路（称为检测电路），并按正常混淆电路流程对其余电路（称为计算电路)进行评估以确定最终结果。如果恶意的 Garbler 构造了错误的电路，在 Evaluator 检查检测电路时可能被发现，也可能最终导致计算电路输出不一致而被 Evaluator 发现。通过适当地选取混淆电路的份数以及检测电路计算电路的划分方法，恶意敌手成功的概率与复制因子s相关。Cut-and-Choose 技术可以分为两类：对整个电路实施 Cut-and-Choose，对电路门实施 Cut-and-Choose。

对电路执行 Cut-and-Choose 技术需由 Garbler 先产生s份混淆电路，Evaluator 产生 Cut-and-Choose 挑战串用于划分检测电路与计算电路。对于检测电路，Garbler 要将每个电路门的输入输出导线所有的标签、混淆表发给 Evaluator，Evaluator 检测该电路实现的函数是否与双方约定计算的函数一致。对于计算电路，按照正常的混淆电路流程来计算。

Nielsen 等人首次在 2009 年提出基于电路门的 Cut-and-Choose 协议，这种方法被称为 LEGO[17]，LEGO 方法要求 Garbler 构造大量与非门发给 Evaluator，Evaluator 随机选择一部分打开检测其正确性，检测通过后，剩余部分被随机置换，并在 Garbler 的帮助下正确拼接成一个容错的 Yao's GC，该容错混淆电路中的每个门都具有冗余的混淆表，称为 bucket。Evaluator 在评估电路时，每个门的输出值取 bucket 中所有混淆表计算结果中占大多数的值。但是由于该方法要用到 Pederson 同态承诺，导致效率较低，Frederiksen 等人在 2013 年提出改进方案 MiniLEGO[18]，采用基于 OT 的异或同态承诺协议，提高了效率。Frederiksen 等人于 2015 年提出的 TinyLEGO 通过对桶的构造进行优化来提高通信效率。Kolesnikov 于 2017 年提出的 DUPLO 中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 等人于 2017 年提出的 JIMU 通过使用 XOR 同态哈希来代替先前的 XOR 同态承诺来提高效率。

2) 3-GC

GC 的参与方一般包括 Garbler 和 Evaluator 两方，当涉及多方（多于两方）时，通常是为了进行验证计算，即验证是否发生了恶意行为。3GC 中的两个 Garbler 记为P_1、P_2，一个 Evaluator 记为P_3。通过两个 Garbler 行为的一致性来发现 Garbler 恶意行为。目前在 3GC 技术方向，有 Choi2014[20]、Mohassel2015[21]等技术方案。Choi 等人在 2014 年提出恶意行为假设下的 3GC，步骤如下：

a) P_1和P_2基于 BMR 生成 GC，并生成输入的 MAC 和 MAC 密钥；
b) P_1和P_2分别发送 GC 给P_3，同时发送自己的输入标签及输入的 MAC；
c) P_1和P_2分别与P_3执行 OT 协议，P_3获得自己的输入标签，然后P_3基于 Cut-and-Choose 验证 GC 的正确性，并验证 MAC 信息的正确性；
d) 最后P_3评估 GC 获得计算结果。

3) PVC-GC

PVC 模型(Public Verifiable Covert，PVC)最早由 Asharov 等人于 2012 年提出[22]，Kolesnikov 等人于 2015 年提出了改进方案[23]，但是以上两种方案效率都较低，无法应用到实际场景中。2017 年阿里巴巴提出并实现了公开可验证方案[24]，主要思想是：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。假设某个参与方实施恶意行为，那么其他参与方可以有λ的概率（λ称为威慑因子，一般>=50%）发现，并将该行为及其签名公开，令恶意参与方承受损失。两个参与方P_1、P_2分别拥有输入v_a、v_b，希望地计算安全的计算函数$f(v_a, v_b)$，以威慑因子$\lambda = 50\%$为例，协议流程为：

a) P_1选择两个随机种子s_1、s_2，P_2和P_1运行 OT 选择一个种子（设P_2选择了s_1）；
b) P_1使用s_1、s_2分别生成 GC1 和 GC2；
c) P_2和P_1之间运行 OT 获取 GC1 中P_2的输入对应的标签；
d) P_2和P_1之间再次运行 OT 获取 GC2 中P_2的输入对应的标签；
e) P_1对 GC1 计算哈希，并把哈希值发给P_2；
f) P_1对 GC2 计算哈希，并把哈希值发给P_2；
g) P_1对上述所有流程进行签名，并把签名发送给P_2；

由于P_2有s_1，因此P_2可以自行生成 GC1，可以自己模拟第 c 步和第 e 步。如果结果与P_1发的不一致，则P_2公布相关签名作为P_1作恶的证据，如果一致就用 GC2 进行真实计算。可见，如果P_1作恶，有 50%的概率被P_2发现，因此理性的P_1会选择不作恶，按照规定执行协议。

对于 PVC，λ越大，发现作恶行为的概率越高，但是计算代价也越大。

4) Auth-GC

Bendlin 等人于 2011 年提出 BDOZ 信息论消息鉴别码（Information-theoretic message authentication code, IT-MAC）[151]。当参与方为P_1、P_2两方时，对秘密x使用加性秘密分享$x = x_1 + x_2$。P_1、P_2分别拥有全局密钥Δ_1、Δ_2，x的份额表示为$[x]$：P_1拥有$[x]_1 = (x_1, M_1, K_1)$，P_2拥有$[x]_2 = (x_2, M_2, K_2)$。K_1、K_2是 MAC 密钥，M_1、M_2是 MAC，满足关系$M_1 = K_2 + \Delta_2 x_1$，$M_2 = K_1 + \Delta_1 x_2$。鉴别P_1的消息x_1时，P_1向P_2提供(x_1, M_1)，P_2使用(K_2, Δ_2)验证$M_1 = K_2 + \Delta_2 x_1$是否成立，鉴别P_2的消息同理。BODZ 鉴别码具有同态特性，秘密y的份额$[y]$为：P_1拥有$[y]_1 = (y_1, M_1', K_1')$、$P_2$拥有$[y]_2 = (x_2, M_2', K_2')$，$P_1$和$P_2$可

以在本地计算$x+y$的份额为$[x+y]$：P_1计算$[x+y]_1=(x_1+y_1,M_1+M_1',K_1+K_1')$、$P_2$计算$[x+y]_2=(x_2+y_2,M_2+M_2',K_2+K_2')$。BDOZ 可以扩展到$n$个参与方的场景，此时每个参与方对于其它$n-1$个参与方的 MAC 密钥都要分别生成一个 MAC，因此存储开销和参与方数量成线性关系。

Xiao Wang 等人在 2017 年提出恒定轮数的可验证混淆电路[25]（Authenticated GC），使用优化的 TinyOT 协议[26]来完成预处理计算。因为恶意混淆器可以通过修改混淆表执行选择失败攻击来推断出评估器的隐私输入，也可以构造错误的混淆表影响评估器的计算结果。为此，该方案引入了预处理函数来构造可以验证的混淆电路。协议可以分为以下四个阶段：

a) 预处理阶段。与计算函数无关的预处理阶段：该预处理阶段生成 IT-MAC 相关的随机数。预处理程序为P_1（混淆器）、P_2（评估器）分别生成全局密钥Δ_1、Δ_2，对于输入导线a的掩码比特λ_a生成 BDOZ 消息鉴别码$[\lambda_a]_1=(\lambda_{a,1},M_{a,1},K_{a,1})$、$[\lambda_a]_2=(\lambda_{a,2},M_{a,2},K_{a,2})$，其中$\lambda_a=\lambda_{a,1}+\lambda_{a,2}$、$M_{a,1}=K_{a,2}+\Delta_2\lambda_{a,1}$、$M_{a,2}=K_{a,1}+\Delta_1\lambda_{a,2}$。同理，对另一根输入导线$b$生成消息鉴别码$[\lambda_b]_1$、$[\lambda_b]_2$。将$[\lambda_a]_1$、$[\lambda_b]_1$发送给$P_1$、将$[\lambda_a]_2$、$[\lambda_b]_2$发送给$P_2$。与计算函数相关的预处理：计算 AND 门时需要预处理程序协助（XOR 在本地按照 Free-XOR 设置来计算），P_1、P_2分别向预处理程序发送$[\lambda_a]_1$、$[\lambda_b]_1$；$[\lambda_a]_2$、$[\lambda_b]_2$。预处理程序计算后得到输出导线掩码比特份额$[\lambda_c]_1$、$[\lambda_c]_2$，其中$s_c \oplus r_c = \lambda_a \wedge \lambda_b$。$P_1$和$P_1$使用掩码比特份额分别生成混淆表的份额。

b) 数据输入阶段：P_1、P_2分别拥有导线a、b的输入，P_1将输入导线b的$(\lambda_{b,1},M_{b,1})$发送给P_2，P_2对 MAC 值验证通过后计算$\lambda_b=\lambda_{b,1}+\lambda_{b,2}$，然后将$\lambda_b \oplus v_b$给$P_1$，$P_1$发送对应的标签给$P_2$。对于输入导线$a$进行类似操作。

c) 电路评估阶段。P_2按照电路拓扑结构物逐层使用输入标签计算输出导线标签。

d) 电路输出阶段。P_2验证P_1的输出导线 MAC 正确后使用标签解密 GC 获得结果。

5) n 方 GC

恶意行为假设下支持任意n个参与方的混淆电路方案主要有两类：一类是 BMR 与其它安全计算技术结合，如 BMR+OT、BMR+SPDZ 等；另一类是将 Xiao Wang 的两方可验证混淆电路扩展到n个参与方[27]。

1990 年提出的 BMR（Beaver-Micali-Rogaway）协议[9]可以将 Yao's GC 扩展到多方情景。首先协议根据需要实现的目标函数生成布尔电路，并向所有参与方公开电路。参与方为$P_1,P_2,\dots,P_n$，对应的输入分别是$x_1,x_2,\dots,x_n \in \{0,1\}$。$P_i$为导线$w$选取长度为$\kappa$的随机比特串 0-seed 和 1-seed：$s_{w,0}^i$、$s_{w,1}^i$，导线$w$的 0-superseed 和 1-superseed 是所有 0-seed 和 1-seed 的拼接：$S_{w,0}=s_{w,0}^1||\dots||s_{w,0}^n$，$S_{w,1}=s_{w,1}^1||\dots||s_{w,1}^n$。如果已知第一个种子对应于 0 且第二个种子对应于 1，会泄露导线真实值，因此为每条导线w选取掩码比特λ_w，阻止各个参与方知道导线的真实值v_w，混淆值记作$\wedge_w = v_w \oplus \lambda_w$（混淆值可以看作是前文提到的选择比特，superseed 可以看作是前文提到的标签）。

a) BMR 生成混淆电路

G：$\{0,1\}^{\kappa} \rightarrow \{0,1\}^{2n\kappa}$是一个随机数发生器，用$G^1$表示G输出的前$L$= nκ位，用$G^2$表示$G$输出的后$L = n\kappa$位。对电路门$g$的混淆是计算$f_g$:$\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$。

输入：

seed：每个参与方P_i输入$s_{x,v}^i$，$x \in \{a,\mathrm{b},\mathrm{c}\}$，$\mathrm{v} \in \{0,1\}$；

扩展 seed：P_i输入L比特$\widetilde{\Upsilon}_{x,v}^i$和$\Upsilon_{x,v}^i$（$\widetilde{\Upsilon}_{x,v}^i = G^1(s_{x,v}^i)$，$\Upsilon_{x,v}^i = G^2(s_{x,v}^i)$），$x \in \{a,b,c\}$，$v \in \{0,1\}$；

掩码比特：P_i输入$\lambda_x^i \in \{0,1\}$，$x \in \{a,b,c\}$。

输出：

对这些输入值进行计算可以得到电路门混淆表，由四个密文组成，每个密文都是对输出导线c的 superseed $S_{c,0}$或$S_{c,1}$进行加密，给定输入a和b的 superseed，可以解密出c的 superseed。具体计算过程如下：

计算$\lambda_x = \oplus_{i=1}^n \lambda_x^i$，$x \in \{a,b,c\}$，计算输出线的 superseed:

$S_{c,A} = (f_g(\lambda_a,\lambda_b) = \lambda_c ?\ S_{c,0} : S_{c,1})$

$S_{c,B} = (f_g(\lambda_a,\overline{\lambda_b}) = \lambda_c ?\ S_{c,0} : S_{c,1})$

$S_{c,C} = (f_g(\overline{\lambda_a},\lambda_b) = \lambda_c ?\ S_{c,0} : S_{c,1})$

$S_{c,D} = (f_g(\overline{\lambda_a},\overline{\lambda_b}) = \lambda_c ?\ S_{c,0} : S_{c,1})$

混淆表的四个密文是：

$A_g = \widetilde{\Upsilon}_{a,0}^1 \oplus \ldots \oplus \widetilde{\Upsilon}_{a,0}^n \oplus \widetilde{\Upsilon}_{b,0}^1 \oplus \ldots \oplus \widetilde{\Upsilon}_{b,0}^n \oplus S_{c,A}$；

$B_g = \Upsilon_{a,0}^1 \oplus \ldots \oplus \Upsilon_{a,0}^n \oplus \widetilde{\Upsilon}_{b,1}^1 \oplus \ldots \oplus \widetilde{\Upsilon}_{b,1}^n \oplus S_{c,B}$；

$C_g = \widetilde{\Upsilon}_{a,1}^1 \oplus \ldots \oplus \widetilde{\Upsilon}_{a,1}^n \oplus \Upsilon_{b,0}^1 \oplus \ldots \oplus \Upsilon_{b,0}^n \oplus S_{c,C}$；

$D_g = \Upsilon_{a,1}^1 \oplus \ldots \oplus \Upsilon_{a,1}^n \oplus \Upsilon_{b,1}^1 \oplus \ldots \oplus \Upsilon_{b,1}^n \oplus S_{c,D}$.

b) BMR 在线阶段

在线阶段，各方仅需为每条电路输入线获得一个 superseed，然后每一方都可以自行评估电路，而无需与其他各方进行交互。

发送和输入有关的混淆值：每个参与方P_i广播和自己输入相关的混淆值$\wedge_w = v_w \oplus \lambda_w$，在该步骤结束时，各方知道每条电路输入线$w$的混淆值$\Lambda w$。

每个参与方P_i广播每个输入导线的一个种子$\wedge_w - seed$，此时参与方知道了每个输入线的$\wedge_w - superseed$。

评估电路：如果$\wedge_a,\wedge_b = (0,0)$，解密$A_g$；如果$\wedge_a,\wedge_b = (0,1)$，解密$B_g$；如果$\wedge_a,\wedge_b = (1,0)$，解密$C_g$；如果$\wedge_a,\wedge_b = (1,1)$，解密$D_g$。解密混淆表密文后，得到输出导线的标签。

表 5 恶意行为假设下 n 方混淆电路比较

协议	技术	每个门的通信开销
SPDZ-BMR	SHE+ZKPoPK	$O(n^4\kappa)$
SHE-BMR	SHE+ZKPoPK	$O(n^3\kappa)$
MASCOT-BMR-FX	OT	$O(n^3\kappa^2)$
OT-BMR	TinyOT	$O(n^2B^2\kappa)$
WRK17b	Optimized TinyOT	$O(n^2B\kappa)$

对恶意行为假设下n方混淆电路比较，如表 5 所示。κ是计算安全参数，B 是 Cut-and-Choose 参数，B 的取值约为 3-5。SPDZ-BMR[28]和 SHE-BMR[29]等基于 SHE 的协议需要用零知识证明协议来证明 SHE 密文是正确加密的，因此开销很大。MASCOT-BMR-FX[30]是对 SPDZ-BMR 的优化，通信开销是n^3，但依然很大。OT-BMR

[31]和 WRK17b[27]都是基于 TinyOT 设计信息论安全的 MAC 来抵抗恶意攻击者，二者效率接近。

3.2.1.3小结

基于对已有混淆电路方案的调研分析，表 6 展示了各个方案的技术特点、安全能力（即敌手行为）及协议支持的参与方数量。在表 7 混淆电路复杂度

协议	生成 GC 计算开销	数据输入开销	混淆表通信开销	评估 GC 计算开销
Yao's GC (Free-XOR、Half gates)	• 全局：随机数生成 1 次 • XOR：生成标签时随机数生成 2 次、异或计算 2 次、Hash 计算 0 次 • AND：随机数生成 2 次、异或计算（包括加密）6 次、Hash 计算 4 次	标签传输：1 次 OT：1 次	XOR：0 个 AND：2 个	XOR：Hash 计算 1 次 AND：Hash 计算 2 次

表 8 中，进一步展示对混淆电路复杂度对比，包括混淆器生成混淆电路时的计算开销、双方输入隐私数据获得输入标签的开销、混淆表的通信开销（密文数量）和评估器解密 GC 的计算开销。

表 6　混淆电路总结

敌手行为	协议名称	参与方数量	技术特点概述
半诚实行为假设	Yao's GC	2	使用输入标签计算出输出标签
	GESS	2	信息论安全的 GC，所需的通信量少于 Yao's GC
	Free-XOR	2	随机选取导线为 0 的标签L_0，导线为 1 的标签$L_1 = L_0 \oplus R$
	Half gates	2	使用一个 XOR 门和两个特殊 AND 门替代原始 AND 门
	Arithmetic-GC	n	将算术电路进行混淆
恶意行为假设	Cut-and-Choose	2	Garbler 生成并发送s份混淆电路；Evaluator 随机检查其中的部分电路，并按正常混淆电路流程对其余电路进行评估以确定最终结果
	3-GC	3	两方基于 BMR 协作生成一个 GC，另外一方验证一致性
	PVC-GC	2	使用数字签名技术
	BMR（+SPDZ/OT）	n	多方协作各生成同一个 GC，彼此验证其一致性
	Auth-GC	2/n	使用优化的 TinyOT 来生成 MAC，构造出可验证的混淆电路

表 7 混淆电路复杂度

协议	生成 GC 计算开销	数据输入开销	混淆表通信开销	评估 GC 计算开销
Yao's GC (Free-XOR、Half gates)	• 全局：随机数生成 1 次 • XOR：生成标签时随机数生成 2 次、异或计算 2 次、Hash 计算 0 次 • AND：随机数生成 2 次、异或计算（包括加密）6 次、Hash 计算 4 次	标签传输：1 次 OT：1 次	XOR：0 个 AND：2 个	XOR：Hash 计算 1 次 AND：Hash 计算 2 次

表 8 混淆电路复杂度（续）

协议	生成 GC 计算开销	数据输入开销	混淆表通信开销	评估 GC 计算开销
GESS	• 输出标签：随机数生成 2 次 • 秘密分享：随机数生成 2 次；异或计算 4 次	标签传输：1 次 OT：1 次	0 个	异或计算 1 次
BMR（N 是参与方数量，n 是比特数）	• 掩码比特生成（随机比特生成，3 次/参与方） • 随机种子生成（随机数生成，6 次/参与方） • 遮盖输入比特（比特异或，2 次/参与方） • 遮盖比特计算（比特异或，3 次/参与方），门电路计算（1 次/参与方） • GC 制作：KDF（4 次/参与方），计算 GC（字符串异或，4*3N 次/参与方	• 输入秘密分享：随机比特生成，(N-1)次/参与方；比特异或(N-1)次/参与方；份额传输，N(N-1)条消息传输 • 遮盖输入比特传输，2N 个 bit 消息；随机种子 2N 条消息	XOR，0 个条；AND，4 个	• 计算遮盖比特（比特异或 2(N-1)次 • KDF（2N 次） 解密 GC（字符串异或，N*n 长度，N 次）
3-GC	• 假设 Cut-and-Choose 中制作的电路个数为 s，其中检查的个数为ρ • 参与方数量为 2 的 BMR GC 制作（s 次）	P_3 输入秘密分享：随机比特生成 1 次；比特异或 1 次；消息 2 条 P_1/P_2：同 BMR	s* BMR GC	• 三方 coin-flipping • GC Check P*4*KDF、P*4*3 次异或；
PVC-GC	• 假设 Cut&choose 中制作的电路个数为 s，其中检查的个数为ρ • 随机种子 OT s 次 • GC 制作 s 次（双方各自制作）	标签传输：s 次 OT：s 次	0	GC 检查 ρ 次，GC 解密 s-ρ次
Auth-GC	• MAC 初始随机数及密钥：F_{pre}，2 次 • GC 制作：	同 BMR	同 BMR	同 BMR

	XOR 门，标签定义，MAC 初始值计算：本地异或 5 次/P_1，4 次/P_2 AND 门，标签定义，MAC 初始值计算：F_{pre}，2 次；本地异或，36 次；KDF，4，本地异或： 16 次			

3.2.2 不经意传输

OT 协议自 1981 年提出后不断发展，协议性能得到优化，协议安全性不断增强。OT 协议是 MPC 协议中最基本的工具，在多种安全计算场景中有着广泛的应用。

3.2.2.1不经意传输性能优化

本节主要介绍 OT 技术性能的优化，介绍经典的基础 OT 方案 NP01，但由于公钥密码实现的 OT 计算效率低，不适合用于大数据的加密和解密，为了解决这个问题研究者们提出了 OT 扩展（OT Extension）技术。OT 扩展是指在基础 OT 阶段使用公钥密码来传输对称密码算法的密钥，在 OT 扩展阶段使用对称密码算法来加密、解密实际数据。

Naor 等人提出的不经意传输协议 NP01[32]是经典的基础 OT 协议，是基于离散对数难题的不经意传输方案，该方案是第一个不需要调用随机预言模型的两轮 OT 协议，此前的方案至少需要三轮交互或者依赖于随机预言模型。

协议的公共参数有有限域Z_q、Z_p和Z_p的生成元g，p和q满足$q|p-1$。发送方的输入是两个长度为l比特的数据(x_0,x_1)，接收方输入一个选择比特r。方案流程如下：

1) 发送方生成两个随机数C和a，C公开，a用来计算计算g^a和C^a。
2) 接收方生成一个随机数k，再生成两个公钥pk_r和pk_{1-r}：$pk_r = g^k$、$pk_{1-r} = \frac{C}{g^k}$，接收方将pk_0发送给发送方。
3) 发送方计算$(pk_0)^a$，以及$(pk_1)^a = C^a/(pk_0)^a$，发送方对发送的数据x_0、x_1进行加密：$E_0 = (g^a, Hash((pk_0)^a) \oplus x_0)$，$E_1 = (g^a, Hash((pk_1)^a) \oplus x_1)$
4) 接收方计算$\mathrm{Hash}((pk_r)^a) = Hash((g^a)^k)$，然后计算$x_r$。
   $E_r \oplus \mathrm{Hash}((pk_r)^a) = (Hash((pk_r)^a) \oplus x_r) \oplus Hash((pk_r)^a) = x_r$

下面以发送方输入$x_0 = 2$、$x_1 = 3$，接收方输入选择比特$r = 1$为例来计算：

1) 发送方生成随机数$\mathrm{C} = 4$，$a = 5$，计算$g^a = g^5$，$C^a = 4^5$，公开C。
2) 接收方选择$k = 6$，计算$pk_1 = g^6$，$pk_0 = \frac{4}{g^6}$。
3) 发送方计算$(pk_0)^a = (\frac{4}{g^6})^5$，$(pk_1)^a = \frac{C^a}{(pk_0)^a} = g^{30}$，加密得到密文：
   $$E_0 = (g^5, Hash((\frac{4}{g^6})^5) \oplus 2)，E_1 = (g^5, Hash(g^{30}) \oplus 3)$$
4) 接收方计算$Hash((g^a)^k) = Hash(g^{30})$，解密$E_1$得到$\mathrm{H}ash(g^{30}) \oplus 3 \oplus Hash(g^{30}) = 3$。

在此方案中，基于离散对数假设，发送方无法求出k，接收方无法求出a，因此无法进一步求出$(pk_0)^a$去解密E_0，只能解密得到自己选择的$x_r = x_1$。若接收方输入选择比特$r = 0$同理可证。

NP01 协议可以从素域推广至椭圆曲线群(例如,SM2 算法定义的椭圆曲线群)；在示例中，使用$Hash((pk_r)^a)$来加密发送数据，也同样可以使用椭圆曲线公钥加密算法（例如，SM2 公钥密码算法）。

Beaver 等人于 1996 年首次提出了 OT 扩展技术[36]，通过运行少量的 OT 协议作为基础 OT,再使用低开销的伪随机数生成算法,来获得大量 OT 执行的效果，但该协议效率很低并不实用。

Ishai 等人在 2003 年提出的 IKNP03[33]是高效的 OT 协议,该方案中运行κ个 OT 来达到执行m个 OT 的效果（$\kappa < m$），其中κ为计算安全参数，运行 OT 时选取的随机字符串长度至少为κ比特。用OT_l^m表示执行m次OT_l（OT_l表示 1-out-of-2 OT 中发送方输入的每个消息为l比特），IKNP03 的基础 OT 阶段是用OT_m^κ（κ次OT_m）生成对称密钥，OT 扩展部分是使用基础 OT 阶段生成的对称密钥进一步实现OT_l^m。

IKNP03 协议的输入包括：发送方 S 输入m对l比特字符串$x_j^0, x_j^1 \in \{0,1\}^l (1 \le j \le m)$，接收方 R 输入选择向量$r = (r_1, \dots, r_m)$；协议输出包括：发送方 S 输出为空，接收方 R 输出为$x_1^{r_1}, \dots, x_m^{r_m}$。

协议分为两个阶段：基础 OT 阶段和 OT 扩展阶段。

1) 基础 OT 阶段

a) S 选择一个随机向量$s \in \{0,1\}^\kappa$，R 选择一个$m \times \kappa$ bit 矩阵$t = [t^1 | \dots | t^\kappa]$，$t^i \in \{0,1\}^m$表示$T$的第$i$列。

b) 双方调用OT_m^κ：R 作为发送方输入$(t^i, t^i \oplus r)$，S 作为接收方输入随机向量s。

c) S 收到$m \times \kappa$矩阵$q = [q^1 | \dots | q^\kappa]$：$q^i = (s^i \cdot r) \oplus t^i$，$q_j = (r_j \cdot s) \oplus t_j$。如图 11，$q^i$是矩阵$q$的列，$q_j$是矩阵q的行。

矩阵q的列：对于$1 \le i \le \kappa$，S 根据自己的s_i来从接收方发送的$(t^i, t^i \oplus r)$中选择一个,如果$s_i = 0, q^i = t^i$,如果$s_i = 1, q^i = t^i \oplus r$，得到列为$q^i = (s_i \cdot r) \oplus t^i$的矩阵q。

矩阵q的行：对于$1 \le j \le m$，当$r_j = 0$时，$q_j = t_j$；当$r_j = 1$时，$q_j = t_j \oplus s$。

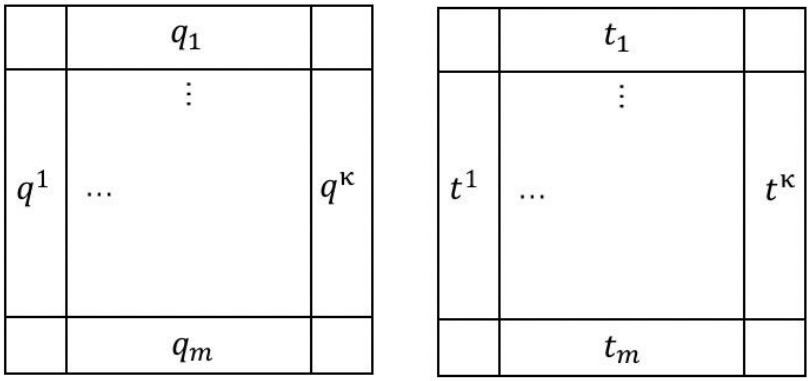

图 11 矩阵 q 和矩阵 t

2) OT 扩展阶段

对于$1 \le j \le m$，

S 发送：$y_j^0 = x_j^0 \oplus H(j, q_j)$，$y_j^1 = x_j^1 \oplus H(j, q_j \oplus s)$；

R 计算：$x_j^{r_j} = y_j^{r_j} \oplus H(j, \mathrm{t_j})$；

其中，H为杂凑算法。

IKNP03 可以抵抗恶意发送方，但是不能抵抗恶意的接收方。恶意接收方可以在基础 OT 阶段的步骤 b)中做出恶意行为，在双方执行OT_m^κ时，恶意接收方发送第i对向量时，令两个向量只在第i个比特不同，一般令第一个向量的第i比特为 0。给定x_i^0，恶意接收方调用两次H可以恢复出s_j。恶意接收方可以利用同样的方法逐比特恢复出s，进而恢复出发送方的$2m$个输入。

2013 年 Kolesnikov 等人将 IKNP03 OT 扩展改进为 1-out-of-n$(n \le \kappa)$OT 扩展[34]，因为传输一个 log n 长比特串的 1-out-of-n 的 OT 等于 log n 个传输 1 比特的 1-out-of-2 的 OT，开销是 IKNP03 的常数倍。2016 年 Kolesnikov 等人在 IKNP03 的基础上进行扩展，实现了任意 n 的 1-out-of-n OT[35]，开销成本几乎等于 IKNP03。2019 年 Boyle 等人提出的 BCG+19b[50]降低了通信开销，可以有效替代 IKNP03。

3.2.2.2不经意传输安全性增强

防止恶意接收方攻击的主要手段是对接收方的输入进行一致性检测（Consistency check）。Nielsen 等人在 2012 年提出了恶意行为假设下的 OT 扩展协议 NNOB12[26]，利用 Hash 函数在基础 OT 阶段进行一致性检测，因为基础 OT 的数量远少于扩展 OT 的数量，因此该方法效率较高，在随机预言模型模型中证明了安全性。Asharov 等人 2015 年提出的 ALSZ15[37]提高了效率，不仅在随机预言模型中证明了安全性，在标准模型中也证明了安全性。Keller 等人提出的 KOS15[30]实现了恶意行为假设下的 1-out-of-2 OT，在随机预言模型中证明了安全性。此外，恶意行为假设下的 1-out-of-n 协议有 OOS16[38]等，恶意行为假设下的 k-out-of-n 协议有 RR16[39]等。

接下来以 ALSZ15 为例，介绍恶意行为假设下的 OT 协议。该方案是在 IKNP03 的基础上进行安全性的增强，发送方在基础 OT 阶段对接收方的选择向量r进行一致性检测，来防止恶意的接收方推测出所有的输入信息x。在基础 OT 阶段，对于发送方的比特串s（长l比特），接收方要作为发送方输入l对消息，每对消息都包含了接收方的选择比特串r。接收方对基础阶段的消息计算哈希并发送给发送方，发送方利用本地消息计算哈希，判断和接收方发送的哈希是否相等，从而判断接收方在不同消息中使用的r是否一致。

发送方的输入：m对字符串(x_j^0, x_j^1)，$1 \le j \le m$。接收方的输入：长度为m的选择向量$r = (r_1, \dots, r_m)$。公共参数包括：对称安全参数κ和统计安全参数ρ，假设$l = \kappa + \rho$。密码学原语：$l \times OT_\kappa$，随机数生成器G：$\{0,1\}^\kappa \to \{0,1\}^{m+\kappa}$。方案具体流程如下：

1) OT 初始化阶段

发送方 S 初始化一个随机向量$s = (s_1, \dots, s_l) \in \{0,1\}^l$，接收方 R 选择$l$对κ比特种子密钥$k_i^0$、$k_i^1$。S 和 R 调用$l \times OT_\kappa$，对于$1 \le \mathrm{i} \le l$，S 作为接收方输入$s$，R 作为发送方输入$(k_i^0, k_i^1)$。

对于$1 \le i \le l$，令$t^i = G(k_i^0)$，$\mathrm{T} = [t^1 | \dots | t^l]$表示$(m+\kappa) \times l$比特矩阵。

2) OT 扩展阶段

a) R 计算$t^i = G(k_i^0)$，$u^i = t^i \oplus G(k_i^1) \oplus r$，向 S 发送$u^i (1 \le i \le l)$。

b) 对r的一致性检查（与半诚实协议相比的主要改变，详细步骤见后文）。

c) 对于$1 \le i \le l$,S 计算$q^i = (s_i \cdot u^i) \oplus G(k_i^{s_i})$，$q^i$作为列组成矩阵 Q。

d) S 计算并向 R 发送：$y_j^0 = x_j^0 \oplus H(j, q_j)$，$y_j^1 = x_j^1 \oplus H(j, q_j \oplus s)$

e) R 计算：$x_j = y_j^{r_j} \oplus H(j, t_j)$。

该协议与半诚实行为假设的协议最大的不同就是对r的一致性检查，在 OT 扩展阶段步骤 a 之后进行。因为有$r^i = t^i \oplus G(k_i^1) \oplus u^i$，对于$r$的第$i$、$j$列，如果$r^i = r^j$，那么则有$u^i \oplus u^j = (t^i \oplus G(k_i^1) \oplus r^i) \oplus (t^j \oplus G(k_j^1) \oplus r^j) = G(k_i^0) \oplus G(k_i^1) \oplus G(k_j^0) \oplus G(k_j^1)$。

在基础阶段，S 只能获得$k_i^{s_i}, k_j^{s_j}$，进一步计算$H(G(k_i^{s_i}) \oplus G(k_j^{s_j}))$。如果$r^i = r^j$，S 计算$H\left(G(k_i^{s_i}) \oplus G\left(k_j^{s_j}\right) \oplus u^i \oplus u^j\right)$等于$H(G(k_i^{\overline{s_i}}) \oplus G\left(k_j^{\overline{s_j}}\right))$，但是 S 不知道$k_i^{\overline{s_i}}$和$k_j^{\overline{s_j}}$（由于 OT 的特性，S 获得$k_i^{s_i}, k_j^{s_j}$，就无法获得$k_i^{\overline{s_i}}$和$k_j^{\overline{s_j}}$）。由于 R 不知道$s_i, s_j$的取值，所以 R 对$s_i, s_j$的 4 种可能的取值都要计算对应的哈希值并发给 S：$h_{i,j}^{0,0} = H\left(G(k_i^0) \oplus G(k_j^0)\right)$、$h_{i,j}^{0,1}$、$h_{i,j}^{1,0}$、$h_{i,j}^{1,1}$，S 对哈希值进行验证后可确定 R 输入的$r$是否一致。

对r的一致性检查具体流程如下：

a) 对于每一对$i, j \subseteq [l]^2$，R 计算 4 个值：

$h_{i,j}^{0,0} = H\left(G(k_i^0) \oplus G(k_j^0)\right), h_{i,j}^{0,1} = H\left(G(k_i^0) \oplus G(k_j^1)\right)$

$h_{i,j}^{1,0} = H\left(G(k_i^1) \oplus G(k_j^0)\right), h_{i,j}^{1,1} = H\left(G(k_i^1) \oplus G(k_j^1)\right)$

发送$H_{i,j} = (h_{i,j}^{0,0}, h_{i,j}^{0,1}, h_{i,j}^{1,0}, h_{i,j}^{1,1})$给$S$。

b) 对于每对$i, j \subseteq [l]^2$，S 知道$s_i, s_j, k_i^{s_i}, k_j^{s_j}, u^i, u^i$，并检查：

i. $h_{i,j}^{s_i, s_j} = H\left(G(k_i^{s_i}) \oplus G\left(k_j^{s_j}\right)\right)$

ii. $h_{i,j}^{\overline{s_i}, \overline{s_j}} = H\left(G(k_i^{s_i}) \oplus G\left(k_j^{s_j}\right) \oplus u^i \oplus u^j\right)$

iii. $u^i \neq u^j$

以上三个等式只要有一个不成立，S 就会退出。

3.2.2.3不经意传输的功能扩展

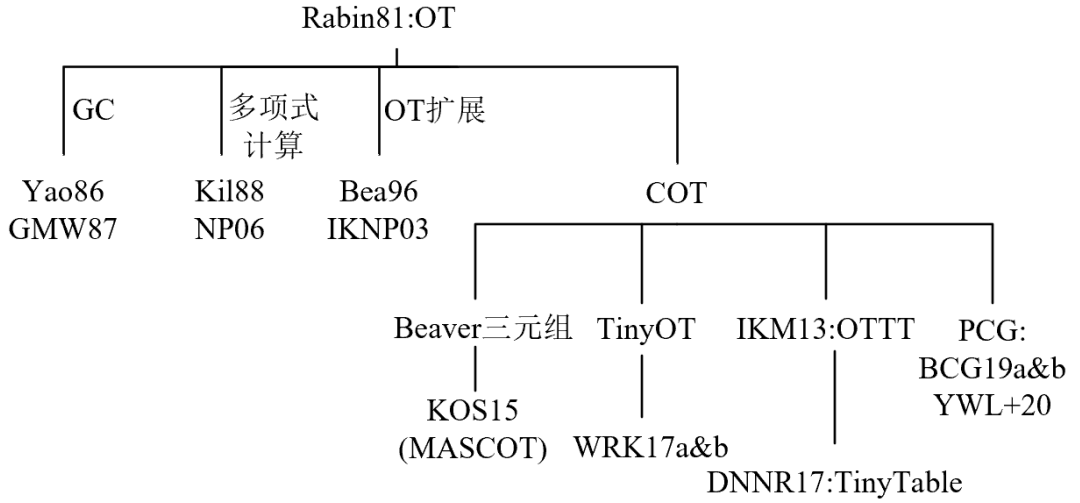

图 12　OT 的功能扩展

OT 是密码学中的一个重要组件，在 MPC 中有着广泛的应用，如图 12 所示：OT 在 Yao's GC 中传输标签、在 GMW 中辅助计算 AND 门；在 Ki188[41]、NP06[42]中可计算多项式函数；OT 自身的效率改进主要是 OT 扩展技术。

COT（Correlated Oblivious Transfer）可以生成具有关系的随机数。COT 协议中，发送方输入r_i、$r_i \oplus \Delta$，接收方输入索引b_i，OT 输出给接收方$r_i \oplus \Delta \cdot b_i$。为了打破传统方法生成和存储 COT 的限制，提出了 PCG（pseudorandom correlation generator）技术，在不破坏安全性的前提下压缩随机数的长度，两个参与方本地生成短的随机数并在本地扩展随机数。Boyle 等人提出的 BCGI18[47]和 BCG+19b[50]将基于 Dual-LPN 的 PCG 用于构造 COT 和 VOLE（vector oblivious linear evaluation），BCGI18 和 BCG+19b 通信轮数多且只能满足半诚实安全模型。Boyle 又提出 BCG+19a[49]在生成分布式密钥时使用（puncturable pseudorandom function, PPRF）代替 BCGII18 和 BCG+19b 中的分布式点函数（distributed point function, DPF），并且通过安全设置 PPRF 密钥将协议改进为恶意安全性，该方案只需要 2 轮通信，通信效率高但计算效率低。Schoppmann 等人的工作 SGRR19[48]基于 primal-LPN 实现了半诚实的 VOLE 协议但没有实现 OT 扩展，该方案计算效率高但通信效率低。Kang Yang 等人于 2020 年提出的 Ferret（YWL+2020）[44]基于 SGRR19 实现了半诚实安全方案，恶意安全方案引入一致性检测来实现，Ferret 与 BCG+19a 相比计算效率高但通信效率低。

COT 的典型应用技术包括：生成 Beaver 三元组，例如 MASCOT[59]；生成消息鉴别码 MAC 的份额，例如 TinyOT[26]、WRK17a&b[25][27]；应用于 OTTT（one-time-true-table）技术，例如 IKM13[45]、DNNR17[46]。

3.2.3 秘密分享

3.2.3.1秘密分享方案-线性运算

本节对常见的秘密分享方案进行介绍，GMW、BGW 是早期经典的秘密分享方案，GMW 是基于 OT 的秘密分享方案，BGW 是基于多项式插值的秘密分享方案。

Sharemind 及其优化方案、$\mathbf{ABY^3}$、SecureNN、PrivPy 都是加性秘密分享方案，其中 Sharemind 优化方案 AFL+2016、$\mathbf{ABY^3}$和 PrivPy 也是门限方案。

1） GMW 协议

GMW（Goldreich-Micali-Wigderson）协议[40]是 Goldreich 等人在 1987 年提出的一种通用、高效的多方安全计算协议。GMW 需要将函数描述为一个布尔电路，电路的每一层布尔门都需要一轮交互。与 Yao's GC 相比，GMW 不需要使用混淆表，因此减少了混淆表传输和加解密操作，节省了计算量和通信量。

GMW 协议支持 n 个参与方进行布尔运算，布尔运算可以最终转化成由 XOR 门和 AND 门组成的二进制电路。首先数据持有方将数据进行比特级的秘密分享，如两个数据持有方将a、b分别进行秘密分享：$a = a_1 \oplus a_2 \cdots \oplus a_n$，$b = b_1 \oplus b_2 \cdots \oplus b_n$，并分别分发给$P_1$、$P_2$、…、$P_n$。协议包含几个步骤：输入秘密分享、评估 XOR 门、评估 AND 门、输出秘密恢复。

如果是“XOR”运算：因为参与方$P_i(i = 1, \ldots, n)$分别拥有份额a_i、b_i，直接在本地计算份额$c_i = a_i \oplus b_i$为异或结果的份额。

如果是“AND”运算：$c = a \cdot b = (\sum_i a_i)(\sum_i b_i) = (a_1 \oplus a_2 \oplus \ldots \oplus a_n) \cdot (b_1 \oplus b_2 \oplus \ldots \oplus b_n) = (\sum_i a_i\, b_i) \oplus \sum_{i \neq j}(a_i\, b_j) = (\sum_i a_i\, b_i) \oplus (\sum_{i<j}(a_i\, b_j \oplus a_j b_i))$，这里$\sum$ 代表连续$\oplus$运算。$a_i b_i$可以在本地计算得出，交叉项$a_i b_j \oplus a_j b_i$需要双方执行 OT 来计算：P_i随机选$c_{i,j}^i$，两方实体使用 1-out-of-4 OT 来使P_j选择正确的$c_{i,j}^j$。P_i以$c_{i,j}^i$、$c_{i,j}^i \oplus a_i$、$c_{i,j}^i \oplus b_i$、$c_{i,j}^i \oplus a_i \oplus b_i$为选择项，$P_j$根据$(a_j, b_j)$为索引来从上述四个密文中选一个作为$c_{i,j}^j$：如果$(a_j, b_j) = (0,0)$，则$c_{i,j}^j = c_{i,j}^i$；如果$(a_j, b_j) = (0,1)$，则$c_{i,j}^j = c_{i,j}^i \oplus a_i$；如果$(a_j, b_j) = (1,0)$，则$c_{i,j}^j = c_{i,j}^i \oplus b_i$；如果$(a_j, b_j) = (1,1)$，则$c_{i,j}^j = c_{i,j}^i \oplus a_i \oplus b_i$，$n$个参与方两两执行上述过程。$P_i$设置自己的份额为$a_i b_i \oplus \sum_{j \neq i} c_{i,j}^i$，$P_j$设置自己的份额为$a_j b_j \oplus \sum_{j \neq i} c_{i,j}^j$。

XOR 门所有计算都会在本地发生，没有网络通信。AND 门由于两两之间需要计算交叉项，计算复杂度为$\mathrm{O}(n^2)$。

Abascal 等人于 2020 年提出了第一个基于 GMW 的两方主动安全协议[51]，使用零知识证明来保证 GMW 计算过程中的正确性。

2） BGW 协议

BGW[52]是 1988 年提出的基于 Shamir 秘密分享的多方安全计算方案。BGW 分别提出了半诚实安全和恶意安全两种协议。

在 Shamir(t,n)秘密分享中，有n个参与方 $P_1 \cdots P_n$分享秘密s。数据分发方 Dealer 首先构造一个多项式$f(x) = a_t x^t + a_{t-1} x^{t-1} + \cdots + ax + s$，秘密为$f(0) = s$，每个参与方$P_i$拥有份额$f(\alpha_i)$。只要有大于等于$t+1$个参与方提交自己的份额$f(\alpha_i)$，可以使用拉格朗日插值多项式快速计算出常数项$s$。

a） 半诚实安全协议

秘密分享：当要对秘密值a、b计算加法和乘法时，基于以上原理，两个数据提供方 Dealer 先选取两个t级多项式 $f_a(x)$、$f_b(x)$对a、b进行秘密分享，将份额分发给n个参与方。

加法：如果要计算加法$a + b$，n个参与方$P_1, \ldots, P_n$直接在本地将份额相加得到和的份额$f_{a+b}(\alpha_i) = f_a(\alpha_i) + f_b(\alpha_i)$。

乘法：如果要计算乘法$a \cdot b$，参与方直接在本地计算积的份额$h(\alpha_i) = f_{a \cdot b}(\alpha_i) = f_a(\alpha_i) \cdot f_b(\alpha_i)$。但是$f_{a \cdot b}(x)$为$2t$次多项式，且其

系数随机性被降低，所以要对份额$f_{a\cdot b}(\alpha_i)$和对应的多项式$f_{a\cdot b}(x)$进行随机化和降阶处理。

随机化处理。经过上述步骤处理后，多项式系数已经不是完全随机了，为了随机化多项式，每个参与方P_i随机选取常数项为 0 的$2t$次多项式$q_i(x)$，将多项式的份额分发给其他参与方。随机化后的多项式为$\tilde{f}_{a,b}(x)=f_{a\cdot b}(x)+\sum_{j=1}^{n-1}q_j(x)$，满足$\tilde{f}_{a,b}(0)=f_{a\cdot b}(0)$，但是$x^i(1\le i\le 2t)$的系数完全随机。然后再进行降阶将多项式次数降为$t$次。

降阶处理。降阶之前多项式的阶数为$2t$，降阶之后多项式的阶数为t。每个参与方P_i计算$c_i=f_a(\alpha_i)\cdot f_b(\alpha_i)$，然后每个$P_i$各自选取次数为$t$次的随机多项式$h_i(x)$，满足$h_i(0)=c_i$、$1\le i\le n$、$t<\frac{n}{2}$。参与方$P_i$向其它参与方分配$c_i$的份额$d_{ij}=h_i(j)$，所有参与方分配结束后，$P_i$拥有信息$d_{1i},d_{2i},\dots,d_{ni}$，此时利用拉格朗日多项式系数$\lambda_1,\dots,\lambda_n$，$P_i$计算$d_i=\sum_{j=1}^{n}\lambda_j d_{ji}$，此时$d_i$为$a\cdot b$的秘密。

在秘密恢复时，使用 Shamir 秘密分享的秘密恢复方法即可恢复出秘密。

b) 恶意安全协议

恶意行为假设下，要求$n\ge 3t+1$，最多允许t个份额错误或者丢失。恶意攻击者可以开展如下攻击行为：在计算阶段发送错误的份额，随机化阶段生成非零多项式等；在输入阶段生成阶数大于t的多项式，导致不同集合的参与方看到不同的多项式；在秘密恢复阶段发送错误的份额。

在秘密恢复阶段，由于会有恶意参与方提供错误的份额，导致恢复出的n个秘密值不一定都在同一个t阶多项式上。BGW 中采用 Reed-solomon 编码$(n,t+1,n-t)$来进行校验。当$t<\frac{n}{3}$时，可以纠正t个错误。例如$t=1$和$n=4$，这样所有的点都位于一条线上，可以查看四个点并找到通过它们的线，来忽略一个已损坏但不在该线上的点。

BGW 采用可验证秘密分享（verifiable secret sharing-VSS）来解决输入的问题：Dealer 要对秘密s进行秘密分享，选取t阶随机多项式$q(x)$，$q(0)=s$；Dealer 选取t阶双向多项式$s(x,y)$（例如：$c_{tt}x^ty^t+\cdots+c_{50}x^5+\cdots$），$s(0,z)=0,s(1,z)=q(z)$；Dealer 向参与方$i$发送$f_i(x)=s(x,\alpha_i)$，$g_i(y)=s(\alpha_i,y)$；每对参与方$i$、$j$检查$f_i(\alpha_j)=g_j(\alpha_i)$，如果不相等参与方会投诉 Dealer。

安全性分析：BGW 有两种安全假设，在半诚实行为假设下，不诚实参与方自己无法计算出敏感信息，即使合谋也恢复不出来敏感信息；在恶意行为假设下，恶意秘密分发方如果发送错误的份额，会被参与方两两使用双向多项式检测出来；如果由于恶意参与方在恢复阶段恢复出错误的秘密值，可以通过 Reed-solomon 检测出错误的恢复值。

3) Sharemind

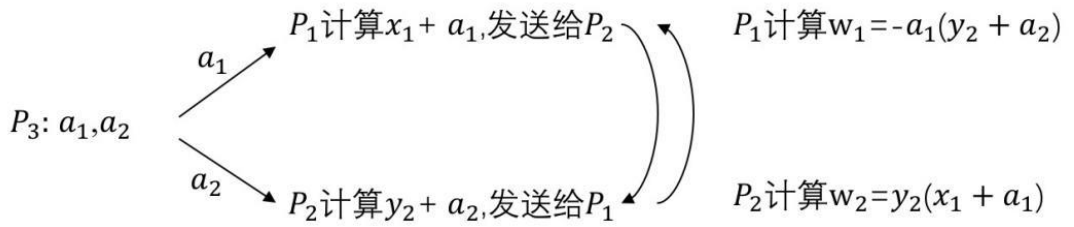

图 13 Sharemind 计算乘法交叉项

Sharemind[53]是 Bogdanov 等人于 2008 年提出的三方秘密分享，支持数据加法和乘法运算。数据提供方分别对x、y进行秘密分享，P_1、P_2、P_3使用各自的份额来实现对数据x、y加法和乘法运算。首先，数据提供方对数据x、y进行秘密分享，随机选取x_1、x_2、y_1、y_2，计算$x_3=x-x_1-x_2$，计算$y_3=y-y_1-y_2$；将x_1、y_1发送给P_1，将x_2、y_2发送给P_2，将x_3、y_3发送给P_3。P_1、P_2、P_3获得相应的份额后，使用各自持有的份额进行数据运算。进行加法运算，计算$z=x+y$，P_1、P_2、P_3在本地将x和y的份额相加得到z的份额。进行乘法运算，计算$z=x\times y$，P_1、P_2、P_3首先对本地份额相乘，得到$x_iy_i(i=1,2,3)$，其它交叉项x_iy_j需要P_1、P_2、P_3交互计算得到，例如计算x_1y_2，如图13所示，P_3生成随机数a_1、a_2，并计算$w_3=a_1\times a_2$。然后P_3将a_1发送给P_1，a_2发送给P_2。P_1计算x_1+a_1并发送给P_2。P_2计算y_2+a_2并发送给P_1。P_1计算$w_1=-a_1(y_2+a_2)$，P_2计算$w_2=y_2(x_1+a_1)$。w_1、w_2、w_3满足$w_1+w_2+w_3=x_1y_2$，其它交叉项可以用相同的方法计算得出，每个参与方P_i将在本地的份额和交叉项份额相加，得到乘法结果z的份额。

4) Sharemind 优化方案 AFL+2016[54]

AFL+2016 是 Araki 等人于 2016 年提出的 2-out-of-3 秘密分享，该方案可以计算布尔电路和算术电路。其中布尔电路支持比特的 XOR 和 AND 运算，算术电路支持环上的加法和乘法运算。

a) 计算布尔电路

三个参与方分别为P_1、P_2、P_3，秘密比特为x被拆分为x_1、x_2、x_3，满足$x_1\oplus x_2\oplus x_3=x$。秘密分发方选取三个随机比特$a_1$、$a_2$、$a_3$，满足$a_1\oplus a_2\oplus a_3=0$，计算$x_1=a_3\oplus x$、$x_2=a_1\oplus x$、$x_3=a_2\oplus x$。$P_1$拥有份额$(a_1,x_1)$、$P_2$拥有份额$(a_2,x_2)$、$P_3$拥有份额$(a_3,x_3)$。任意两个参与方可以恢复出秘密$x$。例如$P_1$提供$a_1$、$P_2$提供$x_2=a_1\oplus x$，可以恢复出$x$。

秘密x的份额是(a_1,x_1)、(a_2,x_2)、(a_3,x_3)；秘密y的份额是(b_1,y_1)、(b_2,y_2)、(b_3,y_3)。对秘密x、y计算 XOR 和 AND 运算。

计算 XOR：$z=x\oplus y$。P_i在本地计算份额$z_i=x_i\oplus y_i$、$c_i=a_i\oplus b_i$。P_1得到$(c_1,z_1)=(a_1\oplus b_1,a_3\oplus b_3\oplus x\oplus y)$，$P_2$得到$(c_2,z_2)=(a_2\oplus b_2,a_1\oplus b_1\oplus x\oplus y)$，$P_3$得到$(c_3,z_3)=(a_3\oplus b_3,a_2\oplus b_2\oplus x\oplus y)$。任意两个参与方可以恢复出$z=x\oplus y$。

计算 AND：$z=x\wedge y$。P_1、P_2、P_3分别拥有随机数α、β、γ，$\alpha\oplus\beta\oplus\gamma=0$。$P_1$计算$r_1=x_1y_1\oplus a_1b_1\oplus\alpha$，向$P_2$发送$r_1$；$P_2$计算$r_2=x_2y_2\oplus a_2b_2\oplus\beta$，向$P_3$发送$r_2$；$P_3$计算$r_3=x_3y_3\oplus a_3b_3\oplus\gamma$，向$P_1$发送$r_3$。然后，$P_1$计算$c_1=r_1\oplus r_3$、$z_1=r_1$，份额为$(c_1,z_1)$；$P_2$

计算$c_2 = r_2 \oplus r_1$、$z_2 = r_2$，份额为(c_2, z_2)；P_3计算$c_3 = r_3 \oplus r_2$、$z_3 = r_3$，份额为(c_3, z_3)。正确性证明：$x_1 y_1 = (a_3 \oplus \mathrm{x})(b_3 \oplus \mathrm{y}) = a_3 b_3 \oplus a_3 \mathrm{y} \oplus x b_3 \oplus xy$，$x_2 y_2 = a_1 b_1 \oplus a_1 \mathrm{y} \oplus x b_1 \oplus xy$，$x_3 y_3 = a_2 b_2 \oplus a_2 y \oplus x b_2 \oplus xy$。因此$z_1 \oplus z_2 \oplus z_3 = r_1 \oplus r_2 \oplus r_3 = (x_1 y_1 \oplus a_1 b_1 \oplus \alpha) \oplus (x_2 y_2 \oplus a_2 b_2 \oplus \beta) \oplus (x_3 y_3 \oplus a_3 b_3 \oplus \gamma) = xy$。所以$P_1$拥有$(c_1, z_1) = (r_1 \oplus r_3, xy \oplus (r_2 \oplus r_3))$，$P_2$拥有$(c_2, z_2) = (r_2 \oplus r_1, xy \oplus (r_1 \oplus r_3))$，$P_3$拥有$(c_3, z_3) = (r_3 \oplus r_2, xy \oplus (r_1 \oplus r_2))$，任意两个参与方可以恢复出$xy$，例如$P_1$提供$r_1 \oplus r_3$，$P_2$提供$xy \oplus (r_1 \oplus r_3)$即可恢复出$xy$。

b) 计算算术电路

秘密分发方选取Z_{2^n}上的随机数a_1、a_2、a_3，满足$a_1 + a_2 + a_3 = 0$。计算秘密x的份额：$x_1 = a_3 - x$、$x_2 = a_1 - x$、$x_3 = a_2 - x$。P_1拥有份额(a_1, x_1)，P_2拥有份额(a_2, x_2)，P_3拥有份额(a_3, x_3)，任意两个参与方可以恢复出秘密x。

秘密x的份额是(a_1, x_1)、(a_2, x_2)、(a_3, x_3)；秘密y的份额是(b_1, y_1)、(b_2, y_2)、(b_3, y_3)。对秘密x、y计算加法和乘法运算。

计算加法：参与方在本地对份额计算，类似前面提到的布尔 XOR。

计算乘法：关系随机数α、β、γ，满足关系$\alpha + \beta + \gamma = 0$ 。

在计算乘法时，首先P_1计算$r_1 = (x_1 y_1 - a_1 b_1 + \alpha)/3$；$P_2$计算$r_2 = (x_2 y_2 - a_2 b_2 + \beta)/3$；$P_3$计算$r_3 = (x_3 y_3 - a_3 b_3 + \gamma)/3$，然后$P_1$、$P_2$、$P_3$进行一次通信，$P_1$将$r_1$发送给$P_2$、$P_2$将$r_2$发送给$P_3$、$P_3$将$r_3$发送给$P_1$。

P_1计算乘法结果秘密分享份额(c_1, z_1)，其中，$c_1 = -r_3 - r_1$、$z_1 = -2r_3 - r_1$；P_2拥有乘法结果份额(c_2, z_2)，其中，$c_2 = r_1 - r_2$、$z_2 = -2r_1 - r_2$；P_3拥有乘法结果份额(c_3, z_3)，其中，$c_3 = r_2 - r_3$、$z_3 = -2r_2 - r_3$。

正确性证明：$x_1 y_1 = (a_3 - x)(b_3 - \mathrm{y}) = a_3 b_3 - a_3 \mathrm{y} - x b_3 + xy$，同理可以计算出$x_2 y_2$、$x_3 y_3$，因此$r_1 + r_2 + r_3 = (x_1 y_1 - a_1 b_1 + \alpha + x_2 y_2 - a_2 b_2 + \beta + x_3 y_3 - a_3 b_3 + \gamma)/3 = xy$。$z_1 = -2r_3 - r_1 = (r_2 - r_3) - (r_1 + r_2 + r_3) = c_3 - xy$，$P_1$的份额$(c_1, z_1) = (c_1, c_3 - xy)$。$P_2$的份额$(c_2, z_2) = (c_2, c_1 - xy)$，$P_3$的份额$(c_3, z_3) = (c_3, c_2 - xy)$，任意两个人可以恢复出$xy$。

5) ABY^3

ABY^3[112]是 Mohassel 等人于 2018 年提出的 2-out-of-3 秘密分享方案，支持数据加法和乘法运算。对数据x、y进行秘密分享，选取随机数x_1、x_2、x_3、y_1、y_2、y_3，满足$x = x_1 + x_2 + x_3$、$y = y_1 + y_2 + y_3$。令P_1拥有份额(x_1, x_2)、(y_1, y_2)，P_2拥有份额(x_2, x_3)、(y_2, y_3)，P_3拥有份额(x_3, x_1)、(y_3, y_1)。

计算加法时，三个参与方分别在本地对份额相加，P_1计算得到$(x_1 + y_1, x_2 + y_2)$、P_2计算得到$(x_2 + y_2, x_3 + y_3)$、P_3计算得到$(x_3 + y_3, x_1 + y_1)$，任意两个参与方可以恢复出$x + y$。

计算乘法时，因为$xy = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = x_1 y_1 + x_1 y_2 + x_1 y_3 + x_2 y_1 + x_2 y_2 + x_2 y_3 + x_3 y_1 + x_3 y_2 + x_3 y_3$，$P_1$在本地计算得到$z_1 = x_1 y_1 + x_1 y_2 + x_2 y_1 + \alpha_1$，$P_2$在本地计算得到$z_2 = x_2 y_2 + x_2 y_3 + x_3 y_2 + \alpha_2$，$P_3$在本地计

算得到$z_3 = x_3y_3 + x_3y_1 + x_1y_3 + \alpha_3$，其中$\alpha_1 + \alpha_2 + \alpha_3 = 0$。参与方$P_i$把积的份额$z_i$发给$P_{i-1}$。此时任意两个参与方可以恢复出$xy$。

6） SecureNN

在 SecureNN[55]中，数据拥有方对其输入数据秘密分享并发送到N个服务器，N个服务器分别使用所持有秘密份额实现对多个数据提供方的数据进行联合训练。该方案中有$N = 3$和4两种设置，数据提供方的数量可以是任意的。SecureNN 实现了矩阵加法、矩阵乘法、比较大小、计算最高有效位（判断正负）、除法等多方安全计算协议。本节主要针对加法和乘法运算进行介绍，其它计算方案在 3.2.3.2 节中介绍。

计算加法时在本地将份额相加，可得到和的份额。乘法的实现分为三方服务器乘法和四方服务器乘法。

SecureNN 中三方服务器乘法的实现采用 Beaver 三元组的思想，区别在于 SecureNN 的输入是矩阵X和Y而不是整数。P_0、P_1是计算方，P_2是辅助计算节点，X为$m \times n$维矩阵，Y为$n \times v$维矩阵，矩阵中的每个元素长度为l比特。协议输入是：P_0拥有的矩阵份额(X_0, Y_0)，P_1拥有的矩阵份额(X_1, Y_1)。公共随机数：0 矩阵U拆分为两个矩阵U_0、U_1，使P_0拥有U_0，P_1拥有U_1。计算过程为：

a） P_2生成随机矩阵A和B，计算得到矩阵$C = A \cdot B$，P_2为P_0生成A_0、B_0、C_0，为P_1生成A_1、B_1、C_1，满足$A_0 + A_1 = A$、$B_0 + B_1 = B$、$C_0 + C_1 = C$。

b） P_0计算$E_0 = X_0 - A_0$，$F_0 = Y_0 - B_0$，P_1计算$E_1 = X_1 - A_1$，$F_1 = Y_1 - B_1$；P_0和P_1通过交换份额恢复出$E = E_0 + E_1$，$F = F_0 + F_1$。

c） 计算矩阵乘积XY的份额：P_0计算$X_0 \cdot F + E \cdot Y_0 + C_0 + U_0$，$P_1$计算$-E \cdot F + X_1 \cdot F + E \cdot Y_1 + C_1 + U_1$。

四方服务器乘法与三方服务器乘法不同，没有使用类似于 Beaver 三元组的计算方法。输入数据X和Y分别被拆分为X_1、X_2；Y_1、Y_2，满足$X_1 + X_2 = X$、$Y_1 + Y_2 = Y$，令P_1拥有X_1、Y_1，P_2拥有X_2、Y_2。随机数V_1、V_2、U_1、U_2满足$V_1 + V_2 = 0$，$U_1 + U_2 = 0$。

a） P_1、P_2在本地计算X_1Y_1、X_2Y_2。

b） P_1把X_1发给P_3，把Y_1发给P_4；P_2把Y_2发给P_3，把X_2发给P_4。

c） P_3计算交叉项$X_1Y_2 + V_1$并发送给P_1；P_4计算$X_2Y_1 + V_2$并发送给P_2。

d） P_1计算积的份额：$X_1Y_1 + X_1Y_2 + V_1 + U_1$；$P_2$计算积的份额：$X_2Y_2 + X_2Y_2 + V_2 + U_2$。

7） PrivPy

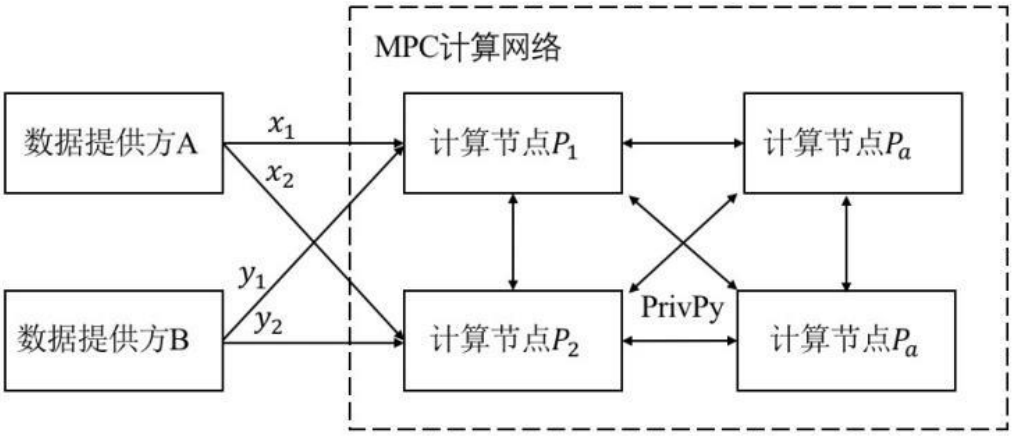

图 14　PrivPy 秘密拆分

PrivPy[56]中实现了 2-out-of-4 秘密分享，四个服务器中只要两个合作就可以恢复出秘密，如图 14 所示。PrivPy 采用了二次秘密分享技术：前两个参与方（记为P_1、P_2）获得第一次份额的结果，然后再计算份额给后两个参与方（记为P_a、P_b）。首先，数据提供方 A 把秘密x拆分为x_1和x_2（$x = x_1 + x_2$）分别发给P_1和P_2，数据提供方 B 把秘密y拆分为y_1和y_2分别发给P_1和P_2。然后，P_1和P_2对秘密份额进行二次秘密分享，P_1和P_2使用随机数r生成$x_1' = x_1 - r_1$、$y_1' = y_1 - r_2$、$x_2' = x_2 + r_1$、$y_2' = y_2 + r_2$。P_1将x_1、y_1发给P_b，将x_1'、y_1'发给P_a；P_2将x_2、y_2发给P_a，将x_2'、y_2'发给P_b。

此时，P_1拥有x_1、y_1、x_1'、y_1'，P_2拥有x_2、y_2、x_2'、y_2'，P_a拥有x_2、y_2、x_1'、y_1'，P_b拥有x_1、y_1、x_2'、y_2'。如果要计算加法，服务器直接在本地对两个加数的份额相加得到和的份额。如要计算乘法$x \cdot y = (x_1 + x_2)(y_1' + y_2') = (x_1' + x_2')(y_1 + y_2)$。首先服务器在本地对份额进行计算：

P_1在本地计算$t_1 = x_1 y_1' - r_{12}$，　$t_1' = x_1' y_1 - r_{12}'$；
P_2在本地计算$t_2 = x_2 y_2' + r_{12}$，　$t_2' = x_2' y_2 + r_{12}'$；
P_a在本地计算$t_a = x_2 y_1' - r_{ab}$，　$t_a' = x_1' y_2 - r_{ab}'$；
P_b在本地计算$t_b = x_1 y_2' + r_{ab}$，　$t_b' = x_2' y_1 + r_{ab}'$。

此时t_1, t_2, t_a, t_b可以恢复出乘法结果，t_1', t_2', t_a', t_b'可以恢复出乘法结果，但是还没达到门限的效果，需要服务器之间交换计算结果：P_1发送t_1给P_b，发送t_1'给P_a；P_2发送t_2给P_a，发送t_2'给P_b；P_a发送t_a给P_2，发送t_a'给P_1；P_b发送t_b给P_1，发送t_b'给P_2。这时任意两个服务器可以恢复出乘法结果，随机数是为了防止单个服务器恢复出x或y。

3.2.3.2秘密分享方案-非线性运算

秘密分享不仅可以用于线性运算，还可以用来计算比较、计算最高有效位、除法等非线性运算，3.2.3.1 节主要针对支持线性运算的方案进行了介绍，本节将进一步针对支持非线性运算的方案进行介绍。

1）ABY^3使用布尔秘密分享执行非线性运算

ABY^3执行线性运算时采用算术秘密分享，执行非线性运算时使用布尔秘密分享，根据计算函数不同需要对份额类型进行转换。算术份额转换为布尔份额需要计算比特分解，即将x转换为比特向量$x[1], \ldots, x[l] \in \{0,1\}$，满足$x =$

$\sum_{i=1}^{k} 2^{i-1}x[i]$。布尔份额在秘密恢复时需要进行比特间的加法，要考虑低位向高位的进位问题，可以使用并行前缀加法器 PPA（Parallel Prefix Adder）实现。比特提取是比特分解的一种特殊情况，只对某个特定比特进行分解。在比较x和y的大小时，可以先计算$t = x - y$，再对t提取最高有效位即可得到x和y的大小。

2）SecureNN 中使用秘密分享计算比较大小、最高有效位

SecureNN 中实现了“私有比较”，即比较l比特秘密x与公共整数r的大小。“私有比较”协议的输入为：P_0、P_1分别拥有x的份额$\{x[i]_0\}_{i\in[l]}$、$\{x[i]_1\}_{i\in[l]}$；公共整数r。计算流程为：对$i = \{l, l-1, \ldots, 1\}$，依次计算$w_i = x[i] \oplus r[i] = x[i] + r[i] - 2x[i]r[i]$，$c_i = r[i] - x[i] + 1 + \sum_{k=i+1}^{l} w_k$。因为$r[i] - x[i] + 1 \geq 0$，所以只要存在$c_i = 0$就说明$\sum_{k=i+1}^{l} w_k = 0$，$r[i] - x[i] = -1$，即高位$i+1$至$l$位$x$的比特和$r$的比特都相等，第$i$位$x[i] > r[i]$，因此$x > r$；若不存在$c_i = 0$，则$x \leq r$。协议输出为$P_2$获得$1\{x > r\}$（1{表达式}中若表达式成立则结果为 1，否则为 0）。SecureNN 中计算最高有效位时，将a的 MSB 计算转换为$y(y = 2a)$的 LSB 计算：$\mathrm{MSB}(a) = \mathrm{LSB}(y)$。计算流程为：$P_0$、$P_1$拥有$Z_{L-1}$上$a$的份额$a_0$、$a_1$，$P_2$选择随机数$x \in Z_{L-1}$，并且将$Z_{L-1}$上$x$的份额和$Z_L$上$x[0]$的份额发送给$P_0$、$P_1$。令$r = y + x \bmod (L-1)$，$P_i(i \in \{0,1\})$在本地计算$r$的份额，双方交换份额可恢复出$r$。因为$r = y + x \bmod (L-1)$，所以$\mathrm{MSB}(a) = \mathrm{LSB}(y) = y[0] = r[0] \oplus x[0] \oplus wrap(y, x, L-1)$。其中 wrap 是一种进位运算，若$y + x \geq L - 1, wrap(y, x, L-1) = 1$，反之同理。可见$\mathrm{wrap}(y, x, L-1) = 1\{x > r\}$（因为如果$x > r$，说明$r$大于$\mathrm{L}-1$且执行了模运算，此时$\mathrm{wrap}(y, x, \mathrm{L}-1)$ 应为 1），可以通过调用“私有比较”函数来计算wrap，然后计算出MSB(a)。

3）CrypTFlow2 使用 SS 和 OT 实现比较大小

CrypTFlow2 中两个参与方使用 OT 来比较大小。P_0、P_1分别拥有l比特长的输入数据x和y，目的是获得$1\{x < y\}$的份额：如果$x < y$，$1\{x < y\} = 1$；如果$x \geq y$，$1\{x < y\} = 0$。假设把比特串x和比特串y分为高位和低位两个块的拼接：$x = x_1 || x_0$，$y = y_1 || y_0$，判断x和y的大小要使用公式 1：$1\{x < y\} = 1\{x_1 < y_1\} \oplus (1\{x_1 = y_1\} \wedge 1\{x_0 < y_0\})$。

按照这个思路如果x和y分解为q个块也可以比较大小：$x = x_{q-1} || \ldots || x_0$，$y = y_{q-1} || \ldots || y_0$，每个块的长度为$m = l/q$比特，把每个块看作是叶子节点，从 0 到$q-1$个叶子节点依次使用 OT 得到 “小于关系树（Inequalities，It）”和“相等关系树（Equalities，Eq）”的份额。对这两个树从叶子节点到根节点，逐层使用公式 1 进行计算，最终得到x和y的大小关系。

CrypTFlow2 中比较大小算法的具体流程如下：

输入：P_0，P_1分别拥有$x \in \{0,1\}^l$，$y \in \{0,1\}^l$

输出：P_0，P_1分别获得$< 1\{x < y\} >_0^B$、$< 1\{x < y\} >_1^B$

a) P_0将输入分解为$x = x_{q-1} || \ldots || x_0$，$P_1$将输入分解为$y = y_{q-1} || \ldots || y_0$，$x_i, y_i \in \{0,1\}^m$。$x$、$y$长$l$比特，将其分别等分为$m$比特的数据块，可以分为$q = l/m$个数据块，每个数据块取值范围是$\{0, \ldots, 2^m - 1\}$。

b) P_0和P_1通过 OT 获得 It 树和 Eq 树叶子节点的份额，具体流程如下：

for $j = \{0, \ldots, q-1\}$：

P_0随机取$< It_{0,j} >_0^B$，$< eq_{0,j} >_0^B \leftarrow \{0, 1\}$

for $k = \{0, \ldots, 2^m - 1\}$ ：

P_0设置$s_{j,k} = < It_{0,j} >_0^B \oplus 1\{x_j < k\}$

P_0设置$t_{j,k} = < eq_{0,j} >_0^B \oplus 1\{x_j = k\}$

end for

P_0, P_1调用$(2^m - 1, 1)$-OT:

输入：P_0作为 Sender 发送$\{s_{j,k}\}_k$，P_1作为Receiver 输入y_j

输出：P_0获得输出$< It_{0,j} >_0^B$，P_1获得输出$< It_{0,j} >_1^B$

同理，对 $t_{j,k}$作类似的操作。

end for

以 It 树为例，上述流程可以理解为对于第j个数据块($j = \{0, \dots, q-1\}$)，P_0选取随机比特$< It_{0,j} >_0^B$，对于$k = \{0, \cdots, 2^m - 1\}$，$P_0$依次计算$s_{j,k} = < It_{0,j} >_0^B \oplus 1\{x_j < k\}$。$P_0$和$P_1$之间执行 1-out-of-$(2^m)$ OT：P_0作为发送方输入$2^m - 1$个$s_{j,k}$，P_1作为接收方输入y_j，OT 执行结束后P_1获得$< It_{0,j} >_1^B$。$< It_{0,j} >_0^B$和$< It_{0,j} >_1^B$满足$< It_{0,j} >_0^B \oplus < It_{0,j} >_1^B = 1\{x_1 < y_1\}$。因为有$q$个数据块，$P_0$和$P_1$执行$q$次 1-out-of-$(2^m - 1)$ OT 后，双方分别获得 It 树的q个叶子节点的份额。Eq 树叶子节点的计算也是相同的原理。

c) It 树和 Eq 树都为二叉树，叶子节点个数为q，树的深度为$log\ q$。将叶子节点层记为第 0 层，根节点层记为第$log\ q$层，那么第i层的节点数量为$q\ /2^i$。计算过程如下：

for $i = \{1, \dots, log\ q\}$ do

for $j = \{0, \dots, (q\ /2^i) - 1\}$ do

对于$b \in \{0,1\}$，p_b调用F_{AND}（F_{AND}的功能：输入是x和y的份额，输出是 $x \wedge y$的份额，x、y是任意数）：输入是$< It_{i-1,2j} >_b^B$和$< eq_{i-1,2j+1} >_b^B$，输出是$< temp >_b^B$（$temp = It_{i-1,2j} \wedge eq_{i-1,2j+1}$）。$p_b$设置$< It_{i,j} >_b^B = < It_{i-1,2j+1} >_b^B \oplus < temp >_b^B$

对于$b \in \{0,1\}$，p_b调用F_{AND}：输入是$< eq_{i-1,2j} >_b^B$和$< eq_{i-1,2j+1} >_b^B$，输出是$< eq_{i,j} >_b^B$

end for

end for

d) 对于 $b \in \{0,1\}$，p_b输出$< It_{log\ q,0} >_b^B$。

$< It_{log\ q,0} >_0^B \oplus < It_{log\ q,0} >_0^B = It_{log\ q,0}$，即$1\{x < y\}$。

4) PrivPy 中的比较大小运算

比较x、y的大小时，首先计算$z = x - y$，提取z的最高有效位c，$c = 1$表示z是负数。将z转换为比特的形式，$z[i]$表示z的第i个比特，$z[1{:}k]$表示z的前k个比特，z的份额z_1、z_2为比特数组，每个份额包含n个比特。计算进位比特时使用公式$c[i+1] = (z_1[i] \oplus c[i]) \wedge (z_2[i] \oplus c[i]) \oplus c[i]$，计算最终的输出比特时使用公式$c[k] = c[k-1] \oplus z_1[k-1] \oplus z_2[k-1]$。可以利用以上方式来提取出两个数差值的最高位，进而获得两个数的大小关系。

3.2.3.3秘密分享安全性增强

SPDZ 系列的协议是恶意安全模型下的秘密分享协议，通过校验 MAC 值来发现恶意行为。原始的 SPDZ 协议开销很大，之后的很多方案来对离线阶段、MAC 值验证等方面进行优化，本节对 SPDZ 及其优化展开介绍。

1) SPDZ

SPDZ[57]是Damgard 等人于2012年提出的一个恶意行为假设下的安全秘密分享协议，用来计算算术秘密分享，容忍$n-1$个主动攻击的参与方，即只要有一个参与方是诚实的而其他参与方都是恶意的，协议会因校验MAC无法通过而中止。SPDZ 是基于加和机制进行秘密分享：$x=x_1+x_2+\cdots+x_n$，$y=y_1+y_2+\cdots+y_n$，并将数据份额分发给n个参与方。对于加法运算，每个参与方P_i在本地计算份额x_i+y_i。对于乘法计算，则通过Beaver 三元组协助完成。

在 SPDZ 预处理阶段，使用 FHE 完成 Beaver 三元组和验证码的生成。SPDZ 中 Beaver 三元组的生成过程为：

a) P_i生成a_i、b_i，$a:=\sum_{i=1}^{n}a_i$，$b:=\sum_{i=1}^{n}b_i$。
b) P_i计算并广播$e_{a_i}=Enc_{pk}(a_i)$，$e_{b_i}=Enc_{pk}(b_i)$。
c) P_i调用$\prod_{ZKPoPK}$来证明生成的密文。
d) 参与方们设置$e_a=e_{a_1}\boxplus\ \ldots\boxplus e_{a_n}$，$e_b=e_{b_1}\boxplus\ldots\boxplus e_{b_n}$。（⊞表示同态加法）
e) 所有参与方计算$e_c\leftarrow e_a\boxtimes e_b$。（⊠表示同态乘法）
f) 参与方们设置$(c_1,\ldots,c_n)\leftarrow Reshare(e_c)$：$P_i$选取$f_i$，$f:=\sum_{i=1}^{n}f_i$；$P_i$计算并广播$e_{f_i}\leftarrow Enc_{pk}(f_i)$；$P_i$调用$\prod_{ZKPoPK}$来证明生成的密文；参与方们计算$e_f=e_{f_1}\boxplus\ldots\boxplus e_{f_n}$，$e_{c+f}=e_c\boxplus e_f$；参与方调用$F_{KEYGENDEC}$来解密$e_{c+f}$获得$c+f$；$P_1$设置$c_1=c+f-f_1$，其他参与方$P_i(i\neq 1)$设置$c_i=-f_i$。

SPDZ 中 MAC 的生成：

	P_1	⋯	P_n	
秘密值	x_1	$+\cdots+$	x_n	$=x$
MAC密钥	α_1	$+\cdots+$	α_n	$=\alpha$
MAC	M_1	$+\cdots+$	M_n	$=M(=\alpha x)$

图 15 SPDZ 的 MAC 值

SPDZ 定义了具备同态性的验证码，如果恶意参与方违反协议会被发现。验证码MAC为$M(x)=\alpha\cdot x$，其中α为MAC全局密钥，x为计算输入数据。如图 15，每个参与方P_i持有密钥α的秘密份额α_i、x的秘密份额x_i和 MAC 值M_i，可以表示为$[\![x]\!]=(x_1,\ldots,x_n,M_1,\ldots,M_n,\alpha_1,\ldots,\alpha_n)$。当打开$[\![x]\!]$时，各参与方首先广播$x_i$并计算$x$，为了确保$x$是正确的，需要承诺并打开$M_i-x\alpha_i$，检查它们的和是否为 0。MAC 值$[\![\cdot]\!]$满足对加法、常数乘法的同态，即以$x$的秘密份额$x_i$执行加法、常数乘法运算的结果，使得 MAC 值在同样执行上述运算后的结果，MAC 算式依然匹配。

SPDZ 中使用一个离线过程准备在线计算时所需的数据，基于 FHE 完成随机数、三元组生成等过程，基于上述离线过程，在线计算中乘法计算的交互次数为2 次广播通信。预处理阶段复杂度为$O(n^2)$。在线阶段总的计算复杂度和n成线性关系。在 SPDZ 的离线阶段中，会使用 ZKPoPK（Zero-Knowledge proof of Plaintext Knowledge）来保证密文产生过程的正确性。ZKPoPK 的作用具体为：

在给定密文C和公钥pk的情况下，用来保证密文提供者确实知晓明文x和随机数r，使得$C = Enc_{pk}(x,r)$，并且x在数值上在正确的明文取值范围之内。

2） SPDZ-2

Damgard 在 2013 年提出的 DKL+2013[58]对 SPDZ 协议进行改进。在原始 SPDZ 协议的在线阶段，需要各个参与方公开全局 MAC 密钥α的份额，才能验证计算结果是否进行，意味着没有被打开的数据份额也不能再使用。该方案在验证的时候不需要恢复出 MAC 密钥，可以继续对已有任何秘密数据进行计算，同时还改进了 ZKPoK 方法。

3） MASCOT

MASCOT[59]是 Keller 于 2016 年提出的 SPDZ 的改进，保持 SPDZ 的在线阶段不变，对离线阶段进行了改进，使用 COT 来生成 Beaver 三元组：发送方的输入是两个随机数x_0、x_1，接收方的输入是选择比特b，执行完 OT 后，接收方获得输出$x_b = -x_0 + b \cdot (x_1 - x_0)$，令$x_1 - x_0 = a$，那么$x_b + x_0 = b \cdot a$。

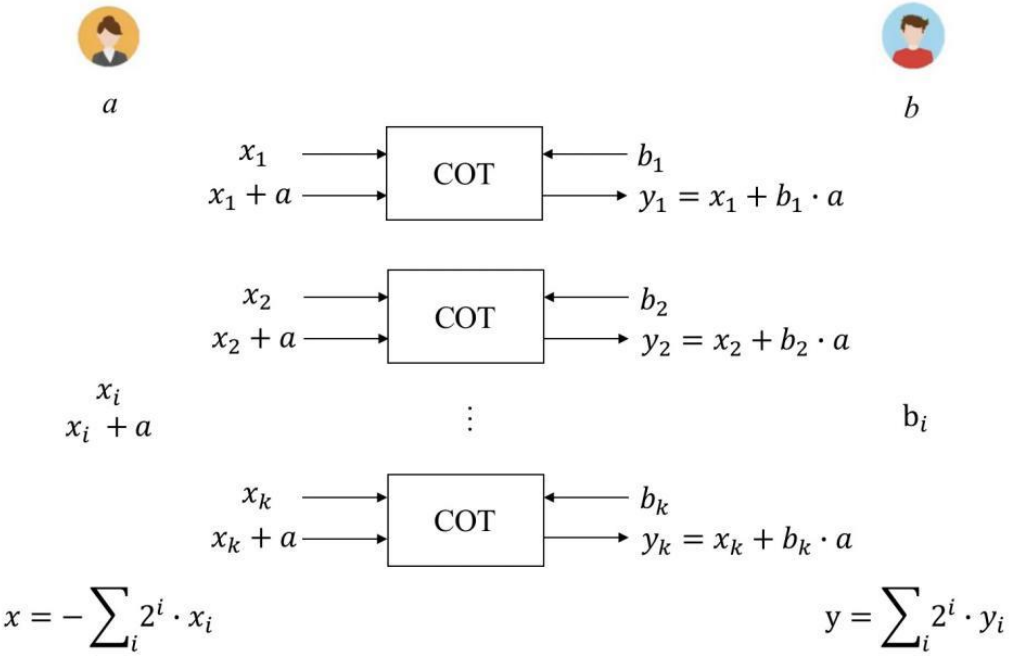

图 16　使用 1-out-of-2 OT 生成 Beaver 三元组份额

因为 Beaver 三元组中a、b、c是长度为k-bit 的算术份额，而执行 OT 时b是选择比特，令$b = \sum_{i=0}^{k} 2^i \cdot b_i$，$b_i \in \{0,1\}$，发送方选取随机数$x_i$，当执行到第$i(1 \le i \le k)$个 OT 时，发送方的输入是$x_i$、$x_i + a$，接收方的输入是$b_i$，接收方获得的输出为$y_i = x_i + b_i \cdot a$。如图 16，执行完$k$次 OT 后，发送方得到$b \cdot a$ 的份额$x = -\sum_i 2^i \cdot x_i$，接收方得到$b \cdot a$ 的份额$y = \sum_i 2^i \cdot y_i = \sum_i (2^i \cdot x_i + 2^i \cdot b_i \cdot a) = -x + ab$。

在恶意行为假设下，如果恶意参与方提供了不一致的输入，可能会导致向量a的比特泄露，为了解决这个问题，参与方选取随机向量r，令$a =< a, r >$、$c =< c, r >$，即计算向量a和r的内积作为a，计算c和r的内积作为c。参与方选取另一个随机向量$\hat{r}$，计算$\hat{a} =< a, \hat{r} >$，$\hat{c} =< c, \hat{r} >$，可以使用$(\hat{a}, b, \hat{c})$来检查(a, b, c)的正确性。选取随机数s，打开$\rho = s \cdot [[a]] - [[\hat{a}]]$，$[[\cdot]]$为前文 SPDZ 中介

绍的 MAC。参与方计算$s\cdot[[c]]-[[\hat{c}]]-[[b]]\rho$，验证计算结果是否为 0，如果为 0 则证明$(a,b,c)$是正确的。MACCOT 通过 OT 实现了该验证过程。

MASCOT 在算术电路上的运行时间比先前方案提高了 200 倍以上。

4) Overdrive

Keller 等人于 2018 年提出了比 MASCOT 性能更好的 KPD2018[60]，在原始的 SPDZ 中需要使用 FHE 才能生成 Beaver 三元组的份额，作者基于 BGV 的加法同态提出 SPDZ 离线阶段的替代方案，由于密文模数更小可以获得更短的密文来减少通信开销，同时也降低了计算开销。与 MASCOT 相比，该方案效率提高了 6-20 倍。

5) $SPDZ_{2^k}$

Cramer 等人于 2018 年提出了基于有限环$(\mathbb{Z}_{2^k})$上的$SPDZ_{2^k}$[61]。该方案使用基于环的加法同态加密框架，在环上选取密钥α，MAC 值αx也是环上的元素，方案效率和有限域上方案一样高效。环上的计算更接近于 CPU 计算单元的实际计算情况，因此有限环上的安全计算具备很高的实际意义。如果基于有限域的计算，直接进行转换，效率会较低。由于在环上存在对 MAC 的攻击方式，研究者对 MAC 计算进行了针对性设计，可以通过扩域方式攻击的可能性降低到可以忽略的程度。该方案的通信开销是 MASCOT 的 2 倍。

以上几个方案表明，针对恶意行为假设，SPDZ 本身实现了 secure with abort 的安全机制，即在参与方存在恶意行为时进行协议退出。还有一种安全机制 secure with identifiable abort，在满足 secure with abort 的前提下，还可以指出存在恶意行为的参与方，此类安全机制的威慑能力更强，例如 BOSV2020[62]。此外，GSZ2020[63]可以使用 secure-with-identifiable-abort 机制先识别出存在恶意行为的参与方，配合应用 Shamir 门限秘密分享技术，在诚实占多数的情况下保证输出计算结果。

3.2.3.4小结

秘密分享自 1979 年提出后不断发展，基于对已有秘密分享方案的调研分析，

表 9 对现有方案进行总结。敌手行为可以分为半诚实行为和恶意行为，参与方数量一般大于等于 3 个。技术原理方面，大多数方案都是用加性秘密分享，只有 BGW 是基于拉格朗日插值多项式。在表 9 中，进一步展示对秘密分享方案复杂度对比，包括数据输入、输出，加法运算、乘法运算、预处理等几个方面。

表 9　秘密分享总结

敌手行为模型	名称	参与方	技术特点概述
半诚实行为假设	GMW	n	使用 1-out-of-4 OT 计算 AND 门的交叉项
	BGW	(t+1)-out-of-n	使用拉格朗日插值法进行秘密分享，使用协同共识的方式进行验证
	Sharemind	3	由一方生成随机数的秘密分享份额，辅助其他两方进行计算
	Sharemind 改进	2-out-of-3	协同生成相关随机数（如 0 的三方秘密分享），辅助进行计算
	ABY^3	2-out-of-3	使用算术秘密分享计算加法和乘法，使用布尔秘密分享来比较大小
	SecureNN	3/4	由辅助节点生成 0 的秘密分享份额，来辅助计算乘法；
	PrivPy	2-out-of-4	通过两个辅助节点进行二次份额计算，避免预处理过程
恶意行为假设	SPDZ	n	使用 Beaver 三元组计算乘法，使用 MAC 保证协议符合恶意安全模型
	SPDZ2	n	不需要恢复出 MAC 密钥即可完整验证
	MASCOT	n	使用 COT 生成 Beaver 三元组
	Overdrive	n	使用 BGV 的加法同态来完成预处理阶段
	SPDZ$\mathbb{Z}_{2^k}$	n	环($\mathbb{Z}_{2^k}$)上的 SPDZ

表 10　秘密分享性能分析

方案	数据输入	加法	乘法	数据输出	预处理
GMW	生成随机比特 n-1 次，比特异或 n-1 次	XOR:1 次异或/参与方	AND：共执行 n(n-1)/2 次 1-out-of-4 OT 每个 OT 生成随机比特 1 次、异或 4 次	比特异或 n-1 次	/
BGW	生成多项式及份额计算	本地加法 1 次/参与方	乘法：1 次/参与方 随机化：生成 0 秘密多项式 1 次/参与方，更新份额计算 n 次加法/参与方。 降阶：每个参与方选取 t 次多项式对乘法结果秘密分享，对份额计算本地乘法（乘以拉格朗日插值系数）n 次；加法（份额相加）n-1 次	拉格朗日插值法	/

表 11　秘密分享性能分析（续）

方案	数据输入	加法	乘法	数据输出	预处理
Sharemind	随机数生成2次，本地加法2次	本地加法1次/参与方	消息交互24条，本地乘法21次，本地加法12次	加法n-1次	生成关系随机数
AFL+2016（Sharemind改进方案）	随机数生成4次，本地加法6次	本地加法1次/参与方	对于布尔（算术）秘密分享，所有参与方共发送消息3次（3次），本地乘法6次（9次），本地加法9次（12次）	加法2次	生成关系随机数
ABY³	随机数生成2次，本地加法2次	本地加法2次/参与方	计算开销：本地乘法9次，本地加法9次 通信开销：协议1轮，消息交互3条	加法2次	生成关系随机数
SecureNN	生成随机比特串1次，比特串异或1次	本地加法1次/参与方	3PC： 计算开销：矩阵乘法5次（m×n维和n×v维）；矩阵加法6次（m×v维）、2次（m×n维）、2次（n×v维） 通信开销：2(2mn+2nv+mv)l比特，协议轮数2轮 4PC： 计算开销：矩阵乘法4次（m×n维和n×v维）、矩阵加法6次（m×v维） 通信开销：2(mn+nv+mv)l比特，协议轮数2轮	加法1次	生成关系随机数
PrivPy	随机数生成2次，本地加法3次；消息交互2条	本地加法2次/参与方	4个参与方（并行）分别：计算本地乘法2次，计算本地加法2次，消息交互1轮（2条消息）	加法1次	两两之间预共享随机数种子，基于种子计算出随机数
SPDZ	基于随机数份额计算的秘密分享：消息交互n次；本地加法n+1次	本地加法：1次/参与方	基于Beaver三元组的乘法实现：（消息交互，2*n*(n-1)次；本地加法5次；本地乘法3次）	加法n-1次	基于SHE的随机数、三元组的秘密份额生成

3.2.4 同态加密

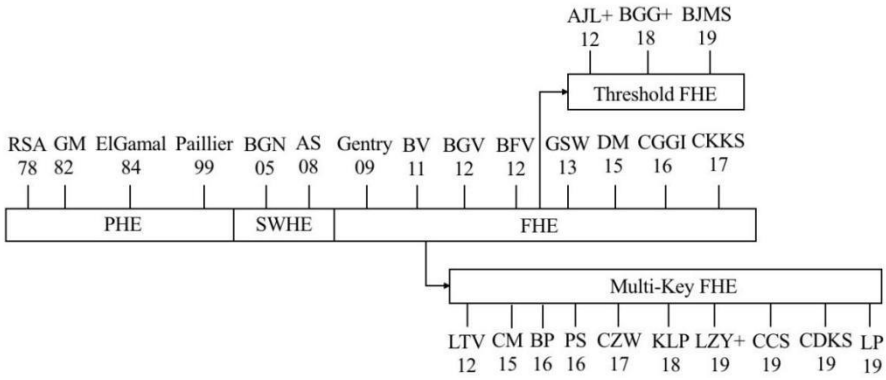

图 17 HE 发展时间线

本节针对同态加密技术的发展情况，对经典及近年来技术现状态进行介绍。图 17 展示了 HE 技术演进和发展的时间轴[64]。HE 加密可以分为两大类：主线是单密钥加密，密文始终在单方拥有的密钥下加密，并使用对应的密钥解密，典型算法有 BGV 等（GM82[65]、AS08[66]、CGGI16[67]）；2012 年起有了多密钥加密这一分支，加解密过程涉及到多个密钥，典型算法有门限同态加密（Threshold HE, ThHE）和多密钥同态加密 （Multi-key HE, MKHE）。

3.2.4.1同态加密

同态加密提出至今，从基于密文可以实现的运算方式的角度，经历了三个阶段[68]，为表 13 中所示：1978—1999 年是部分同态加密的繁荣发展时期；1996—2009 年是部分同态加密与浅同态加密的交织发展时期，也是浅同态加密的繁荣发展时期；2009 年以后是全同态加密的繁荣发展时期。部分同态加密按照明文空间上能实现的代数或算术运算分为乘法同态和加法同态等，常见的算法有 ElGamal 和 Paillier。浅同态加密能同时进行有限次乘法和加法运算的加密，例如 Boneh 等人提出的加密 BGN05[69]，可以执行一次乘法和若干次加法运算。全同态加密可以进行任意次数的加法和乘法，按照构造思想大致可以分为 3 代：以 Gentry 设计为代表的、基于格上困难问题构造的第 1 代全同态加密方案[70]；以 Brakerski-Vaikuntanathan 为代表的、基于 LWE 或 RLWE 困难问题构造的第 2 代全同态加密；以 Gentry-Sahai-Waters 为代表的、基于 LWE 或 RLWE 困难问题构造的第 3 代全同态加密。

表 12 同态加密分类

类型		算法	时间	说明	实际应用
部分同态加密	乘法同态	RSA	1977	非随机化加密，具有乘法同态性的原始算法面临选择明文攻击	在非同态场景中应用广泛
		ElGamal	1985	随机化加密	DSS 数字签名标准基于 ElGamal 数字签名算法的变体
	加法同态	Paillier	1999	应用最为成熟	联合学习

表 13　同态加密分类（续）

类型	算法	时间	说明	实际应用
浅同态加密(有限次数全同态)	Boneh-Goh-Nissim	2005	仅支持1次乘法同态运算和加法同态运算	/
全同态加密	Gentry	2009	第一代全同态加密，性能较差	/
	BGV	2012	第二代全同态加密，性能相对较好	IBM HElib 开源库
	BFV	2012	第二代全同态加密，与BGV 类似	MicrosoftSEAL 开源库
	GSW	2013	第三代全同态加密，基于近似特征向量	TFHE 开源库
	CKKS	2017	可实现浮点数近似计算，适合机器学习建模场景	HElib、SEAL、HEAAN

以下为几种典型的同态加密算法：

1）BGV

BGV[71]是一种 leveled HE，允许任意数量的加法和有限次的连续乘法。该算法在使用自举时可以计算任意次的乘法，但是计算开销很大并不实用。BGV 在进行同态加法和同态乘法之后噪声增长，用模数变换来降噪；同态乘法之后密文和密钥维度都会扩张为原来的平方，用密钥置换技术来约减密文维数。BGV 由以下步骤组成：

$\mathrm{BGV.Setup}(1^{\lambda},1^{L})$：给定安全参数$\lambda$，乘法深度$L$，选择一个环多项式$\phi(x)=x^{d}+1$，$d$是 2 的幂，$R=\mathbb{Z}[x]/(\phi(x))$是$d$阶多项式环。在环$R$上选取噪声分布$\chi$和密钥分布$\Psi$，密文模数$q$和明文模数$t$，输出公共参数$pp=(R,\chi,\Psi,q,t)$。

$\mathrm{BGV.KeyGen}(pp)$：选取私钥$s\leftarrow\Psi$，噪声$e\leftarrow\chi$，选取随机值$a\leftarrow R_q$，设置私钥$sk=s$，公钥$pk=(b,a)\in R_q^2$，其中$b=-as+te(mod\ q)$。

$\mathrm{BGV.Enc}(pk,m)$：给定明文消息$m\in R_t$，公钥$pk=(b,a)$，选取随机数$r\leftarrow\Psi$，噪音$e_0,e_1\leftarrow\chi$，加密获得密文$c=(c_0,c_1)\in R_q^2$，$c_0=rb+m+te_0$，$c_1=ra+te_1$。

$\mathrm{BGV.Dec}(sk,c)$：给定密文$c=(c_0,c_1)\in R_q^2$和私钥$sk=s$，令$s=(1,s)\in R_q^2$，$m=(<c,s>(mod\ q))(mod\ t)\in R_t$。

$\mathrm{BGV.Add}(c,c')$：给定密文 c=(c_0,c_1)、$c'=(c_0',c_1')$，计算同态加法得到$c_{add}=c+c'=(c_0+c_0',c_1+c_1')\in R_q^2$。

$\mathrm{BGV.Mult}(c,c')$：给定密文$c=(c_0,c_1)$、$c'=(c_0',c_1')$，计算同态乘法得到：$c_{Mult}=c\otimes c'=(\widetilde{c_0},\widetilde{c_1},\widetilde{c_2})$，解密时需要$s\otimes s$来解密。对密文计算张量积之后，得到的新密文、新密钥维度是原来的平方。

BGV 采用密钥置换技术控制密文向量的维数膨胀，采用模数变换技术控制密文同态运算产生的噪声增长。

a)　密钥置换 Key Switching(对乘法密文降维)

用同一个密钥s对消息m_1、m_2加密得到两个密文c_1、c_2，对两个密文计算同态乘法得到：$c = c_1 \otimes c_2 = Enc_{s\otimes s}(m_1 \cdot m_2)$，密文和密钥的维度都变成了原来的平方。所以要使用 Key Switching 来对密钥和密文降维，将密文$c = Enc_{s_1}(m_1 \cdot m_2)$转换为$c' = Enc_{s_2}(m_1 \cdot m_2)$。需要用到的两种运算：$\mathrm{BitDecomp}(x \in R_q^n, q)$运算将$x$分解成二进制表示形式$x = \sum_{j=0}^{\lfloor \log q \rfloor} 2^j \cdot u_j$，$u_j \in R_2^n$，输出为$(u_0, u_1, \dots, u_{\lfloor \log q \rfloor}) \in R_2^{n\cdot\lfloor \log q \rfloor}$；$\mathrm{Powersof2}\left(x \in R_q^n, q\right)$运算输出为$\left(x, 2\cdot x, \dots, 2^{\lfloor \log q \rfloor}\cdot x\right) \in R_q^{n\cdot\lfloor \log q \rfloor}$。

密钥置换过程分为两个步骤：置换密钥生成和密钥置换。

置换密钥生成$\mathrm{SwitchKeyGen}(s_1 \in R_q^{n_1}, s_2 \in R_q^{n_2})$：运行$\mathrm{A} \leftarrow \mathrm{E.PublicKeyGen}(s_2, N)$，$N = n_1 \cdot \lceil \log q \rceil$（这里$n_1 = (n+1)^2$），将$\mathrm{powersof2}(s_1) \in R_q^N$加到$A$的第一列获得矩阵$B$。输出置换密钥（也叫评估密钥$ek$）$\tau_{s_1 \to s_2} = B$。

密钥置换$\mathrm{SwitchKey}\left(\tau_{s_1\to s_2}, c_1\right)$：输出$c_2 = BitDecomp(c_1)^T \cdot B \in R_q^{n_2}$。

b) 模数变换 Modulus Switching（降低噪声）

一个不知道私钥s的评估者可以将模q的密文c，转换为另一个模p的密文c'，使得即$[< c', s >]_p = [< c, s >]_q \ mod\ 2$。从$c$到$c'$的转换只包含$(\frac{p}{q})$倍的放缩和适当的近似舍入：$p$、$q$两个模数都是奇数，对$c$乘$(\frac{p}{q})$得到密文再取近似，使得最终的密文$c' = c \ mod\ 2$。

2） BFV

BFV 是基于 RLWE 难题的全同态加密算法，来源于论文 Bra12[72]和 FV12[73]，与 BGV 类似可以实现 leveled HE，使用自举可以实现任意次乘法但计算开销大。

$\mathrm{BFV.Setup}\left(1^\lambda, 1^L\right)$：给定安全参数$\lambda$，乘法深度$L$，选择一个环多项式$\phi(x) = x^d + 1$，$d$是 2 的幂，$\mathrm{R} = \mathrm{Z[x]}/(\phi(x))$是$d$阶多项式环。在环$R$上选取噪声分布$\chi$和密钥分布$\Psi$，密文模数$q$和明文模数$t$，输出公共参数$pp = (R, \chi, \Psi, q, t)$。

$\mathrm{BFV.KeyGen}(pp)$：给定公共参数p，选取$s \leftarrow \Psi$，选取随机值$a \leftarrow R_q$，选取噪声$e \leftarrow \chi$，令私钥$sk = s$，公钥$pk = (b, a) \in R_q^2$，$b = -as + e\ (mod\ q)$。

$\mathrm{BFV.Enc}(pk, m)$：给定公钥$pk = (b, a)$，消息$m \in R_t$，选取随机数$r \leftarrow \Psi$，噪音$e_0, e_1 \leftarrow \chi$，使用放缩因子Δ计算放缩消息$\Delta m = \left\lfloor \frac{q}{t} \right\rfloor m$，加密得到密文$c = (c_0, c_1) \in R_q^2$，其中$c_0 = rb + e_0 + \Delta\mathrm{m}$，$c_1 = ra + e_1$。

$\mathrm{BFV.Dec}(sk, c)$：给定密文$c = (c_0, c_1) \in R_q^2$和密钥$s = (1, s)$，解密计算$\lfloor (t/q) < \mathrm{c, s} > \rceil \in R_t$。

$\mathrm{BFV.Add}(c, c')$：两个密文$c = (c_0, c_1), c' = (c_0', c_1')$，计算$c_{add} = (c_0 + c_0', c_1 + c_1')\ (mod\ q)$。

$\mathrm{BFV.Mult}(c, c')$：将两个密文$\mathrm{c}, c' \in R_q^2$相乘，得到密文$\tilde{c}_{mult} = (\widetilde{c_0}, \widetilde{c_1}, \widetilde{c_2}) \in R_q^3$，其中$\widetilde{c_0} = \lfloor (\mathrm{t/q}) \cdot (c_0 c_0') \rceil$，$\widetilde{c_1} = \lfloor (\mathrm{t/q}) \cdot (c_0 c_1' + c_1 c_0') \rceil$，$\widetilde{c_2} = \lfloor (\mathrm{t/q}) \cdot (c_1 c_1') \rceil$。得到的密文是用密钥$s^2$加密的，使用密钥置换技术转换为密钥 s 加密的密文$c_{mult} = KeySwitch(\tilde{c}_{mult}, ek)$。

3）CKKS

CKKS（HEAAN）[74]是由 Cheon 等人于 2017 年提出的支持对浮点数做近似计算的同态加密算法。CKKS 算法在执行加密算法之前，先执行编码算法将复数或实数向量映射到环 R 的多项式上：选择一个环多项式$\phi(x)=x^d+1$，$R=\mathbb{Z}[x]/(\phi(x))$是$d$阶多项式环，编码过程是将$\frac{d}{2}$个消息映射到多项式$m'$上，$m'\in R$。如果向量中的消息数量小于$\frac{d}{2}$，则剩余的槽设置为 0。$m'$中另一半槽会被消息的共轭值填充。然后使用缩放因子$\Delta$将$m'$放大得到的编码输出为明文$m$。同样地，CKKS 使用在解密后得到明文多项式，使用与编码过程相反的步骤对明文多项式解码得到复数或实数向量。

$\mathrm{CKKS.Setup}(1^\lambda,1^L)$：给定安全参数$\lambda$，乘法深度$L$，选择一个环多项式$\phi(x)=x^d+1$，$d$ 是 2 的幂，$R=\mathbb{Z}[x]/(\phi(x))$是$d$阶多项式环。在环$R$上选取上界为$\mathcal{B}$的噪声分布$\chi$和密钥分布$\Psi$，密文模数为 q，选取$a\leftarrow R_q$，输出公共参数$pp=(R,\chi,\Psi,\mathcal{B},q,a)$。

$\mathrm{CKKS.KeyGen}(pp)$：给定公共参数pp，选取$s\leftarrow\Psi$，选取噪声$e\leftarrow\chi$，选取$a\leftarrow R_q$，令私钥$sk=s$，公钥$pk=(b,a)\in R_q^2$，$b\leftarrow -as+e(mod\ q)$。

$\mathrm{CKKS.Enc}(pk,m)$：给定公钥pk和明文m，选取$r\leftarrow\Psi$，e_0、$e_1\leftarrow\chi$，加密后密文为$c=(c_0,c_1)\in R_q^2$，其中$c_0=rb+e_0+m$，$c_1=ra+e_1$。

$\mathrm{CKKS.Dec}(sk,c)$：给定密文$c=(c_0,c_1)\in R_q^2$和私钥$sk=s$，令$s=(1,s)$，解密得到$m=<c,s>(mod\ q)$。

$\mathrm{CKKS.Add}(c,c')$：对于两个密文$c=(c_0,c_1)$和$c'=(c_0',c_1')$，计算同态加法得到$c_{add}=(c_0+c_0',c_1+c_1')$。

$\mathrm{CKKS.Mult}(c,c')$：对于两个密文$c=(c_0,c_1)\in R_q^2$和$c'=(c_0',c_1')\in R_q^2$，计算同态乘法得到被密钥$(1,s,s^2)$加密的密文$\tilde{c}_{mult}=(\widetilde{c_0},\widetilde{c_1},\widetilde{c_2})\in R_q^3$，使用置换密钥$ek$将$\tilde{c}_{mult}$转换为被密钥 s 加密的密文$c_{mult}=(c_0,c_1)\in R_q^2$。

$\mathrm{CKKS.Rescale}\ (c)$：计算$c'=\left\lfloor\frac{q'}{q}\cdot c\right\rceil\in R_{q'}^2$，完成对密文的降噪。

CKKS 支持针对实数或复数的浮点数加法和乘法同态运算，得到的计算结果为近似值，适用于机器学习模型训练等不需要精确结果的场景。

3.2.4.2多方同态加密

现阶段，多密钥同态加密有两类：门限 FHE(Threshold-FHE)和多密钥 FHE(Multikey-FHE)，如图 18 所示。

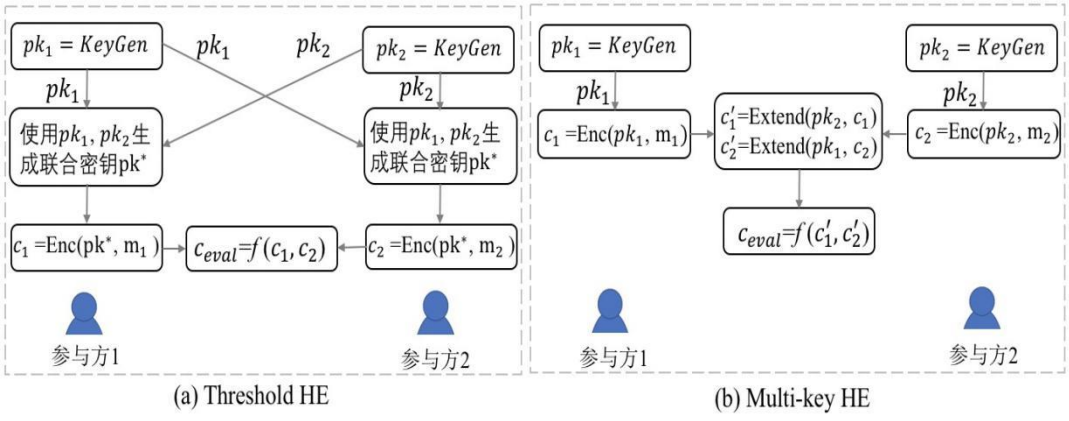

图 18　门限 HE 和多密钥 HE 的加密步骤

在 Threshold-FHE（ThHE）门限同态加密算法中，参与方预先用各自的公钥生成联合公钥pk^*，并用联合公钥加密明文得到密文，对密文进行同态评估，必须互相配合才能解密。Multikey-FHE（MKHE）对不同公钥加密的密文进行同态评估。

1） 基于 Threshold-FHE 的 MPC 协议

Threshold-FHE 有三个组成部分：分布式密钥设置、同态加密函数、分布式解密协议。将通用 Threshold-FHE 定义为 PPT 算法基本元组 $\mathrm{ThHE} = (\mathrm{Setup}, \mathrm{JointKeyGen}, \mathrm{Enc}, \mathrm{Eval}, \mathrm{Dec})$：

$\mathrm{ThHE.Setup}(1^\lambda) \to ((pk_1, sk_1) \dots (pk_n, sk_n))$：给定安全参数$\lambda$，输出$n$个密钥对$(pk_i, sk_i)$。

$\mathrm{ThHE.JointKeyGen}(pk_1, \dots, pk_n) \to pk^*, ek^*$：给定$n$对公钥$(pk_1, \dots, pk_n)$，交互式算法输出联合公钥$pk^*$、评估密钥$ek^*$。

$\mathrm{ThHE.Enc}(pk^*, m) \to c$：给定联合公钥$pk^*$和消息$m$，加密算法输出密文$c$。

$\mathrm{ThHE.Eval}(pk^*, f, c, c') \to c_{eval}$：给定联合公钥$pk^*$和两个密文$c$、$c'$，评估算法输出评估密文$c_{eval} = f(\mathrm{c}, c')$。

$\mathrm{ThHE.PartDec}(sk_i, c) \to \rho_i$：给定一个$pk^*$加密的密文$c$，和一个私钥的份额$sk_i$，执行部分解密算法并输出部分解密明文$\rho_i$。

$\mathrm{ThHE.Combine}(\rho_1, \dots, \rho_T) \to m$：给定一系列部分解密明文$(\rho_1, \dots, \rho_T)$，组合这些部分明文并执行最终的解密获得明文$m$。

ElGamal 和 Paillier 都可以扩展成门限加密的形式，以 ElGamal 为例，n个参与方中每个参与方i拥有密钥$pk_i = g^{s_i}$，$sk_i = s_i$，联合密钥为$pk^* = \prod_{i=1}^n pk_i = g^{\sum_{i=1}^n s_i} = g^{s^*}$，使用公钥$pk^*$和随机数$r$对明文加密：$c = (c_0, c_1) = (g^r, m \cdot (g^{s^*})^r)$。解密时每个参与方$i$计算$\rho_i = (c_0)^{-s_i}$，$\sum_{i=1}^n \rho_i = g^{-rs^*}$。每个参与方$i$轮流用自己的密钥对密文来进行部分解密$\hat{c}_1 \cdot \rho_i = m \cdot g^{rs^*} \cdot g^{-rs_i}$，所有参与方都解密之后得到明文$m$。

Asharov 等人提出了 BGV 的门限加密算法 AJL+12[75]，流程如下：

$\mathrm{ThBGV.Setup}(1^\lambda, 1^L)$：安全参数为$\lambda$，乘法深度为$L$，运行 BGV.Setup 获得公共参数$pp = (R, \chi, \Psi, q, t, a)$，$a$是 CRS 参数。

$\mathrm{ThBGV.KeyGen}(pp) \to ((pk_1, sk_1), \dots, (pk_n, sk_n))$：给定公钥参数$pp$，选取$n$个私钥$s_i$和误差$e_i$，$i \in \{1, \dots, n\}$，输出密钥对$(pk_i, sk_i)$，其中$pk_i = (as_i + te_i, a)$，$sk_i = s_i$。

$\mathrm{ThBGV.JointKeyGen}(pk_1, \dots, pk_n) \to pk^*$，$ek^*$：给定$n$对公钥$(pk_1, \dots, pk_n)$，计算联合密钥$pk^* = (\sum(as_i + te_i), a)$，计算评估密钥$ek^* = \{\mathrm{Enc}(pk^*, \sum s^*[\mathrm{i}]s^*[\mathrm{j}])\}$。

$\mathrm{ThBGV.Enc}(pk^*, m) \to c$：给定联合密钥$pk^*$和消息$m$，加密得到密文$c = (c_0, c_1)$，其中$c_0 = r(as^* + te^*) + m$，$c_1 = ra$。

$\mathrm{ThBGV.Eval}(pk^*, f, c, c') \to c_{eval}$：给定联合密钥$pk^*$和两个密文$c$、$c'$，评估算法输出评估密文$c_{eval} = f(\mathrm{c}, c')$，评估算法可以是加法或者乘法，如果是乘法可以通过评估密钥ek^*降维。

$\mathrm{ThBGV.PartDec}(sk_i, c) \to \rho_i$：给定使用$pk^*$加密的密文$c$，和私钥的份额$sk_i$，计算部分解密$\rho_i = (c_1 s_i + te_i')$。

$\mathrm{ThBGV.Combine}(\rho_1,\dots,\rho_n,c)\to m$：给定部分解密结果$(\rho_1,\dots,\rho_n)$，解密密文$(c_0-\sum\rho_i)=(c_0-c_1\sum s_i)=(c_0-c_1s^*)=\widetilde{m}\ mod\ t=m$。在密钥置换阶段需要使用评估密钥$ek^*$来使密文再次和$s^*$成线性关系。

Threshold-FHE 有两种安全性假设：不诚实占多数，诚实占多数。在不诚实占多数的安全假设下可以构造 n-out-of-n 门限方案，前文介绍的 ElGamal 和 BGV 的门限方案都是基于不诚实占多数的安全假设。诚实占多数安全性假设下的方案有：Desmedt 等人提出的 ElGamal 的门限版本[76]，其中 n 个用户中只需要 t 个用户即可解密，该方案利用了 Shamir 的线性秘密分享。Pedersen[77]通过在密钥设置时不再需要可信任的分发方来改进该方案，可以对秘密份额进行验证，但是解密仍然需要可信的分发方来恢复多项式并进行解密。Boneh 等人[78]提出第一个基于 LWE 问题的（t-out-of-n）ThHE。此外，门限 HE 还有 BGG+18[80]和 BJMS19[81]。

2） 基于 Multikey-FHE 的 MPC 协议

与 ThHE 相比，MKHE 无需生成联合密钥，参与方可以将不同密钥加密的密文扩展到串联密钥加密的密文。与 ThHE 相同，在解密阶段需要各参与方合作解密。Multikey-FHE 可以分为三类：基于洋葱加密（onion encryption）实现，单跳(Single-hop)MKHE，多跳（Multi-hop）MKHE，如果扩展密文在同态评估后不能进一步扩展到其它密钥则归类为单跳 MKHE，否则归类为多跳 MKHE。

MKHE 是由多项式时间算法组成的密码，$\mathrm{MKHE}=(\mathrm{Setup,KeyGen,Enc,Extend,EvalKeyGen,Eval,Dec})$：

$\mathrm{MKHE.Setup}\left(1^\lambda,1^\kappa\right)\to pp$：给定一个安全参数$\lambda$，和密钥数量的界限$\kappa$，输出公共参数$pp=(R,\chi,\Psi,B,q,t,a)$。

$\mathrm{MKHE.KeyGen}(pp)\to(pk,sk)$：给定公共参数$pp$，密钥生成算法输出公钥$pk$，私钥$sk$。

$\mathrm{MKHE.Enc}(pk,m)\to c$：给定公钥$pk$和消息$m$，加密算法输出密文$c$。

$\mathrm{MKHE.Extend}(\{pk_1,\dots,pk_k\},c)\to\bar{c}$：给定$k$个公钥$pk_1,\dots,pk_k$，$k<\kappa$，密文$c$，输出是拼接公钥$\overline{pk}$加密下的密文$\bar{c}$。

$\mathrm{MKHE.EvalKeyGen}\left(\overline{pk}\right)\to\overline{ek}$：给定拼接公钥$\overline{pk}$，生成对应的评估密钥$\overline{ek}$。

$\mathrm{MKHE.Eval}\left(\overline{pk},f,\bar{c},\bar{c}'\right)\to c_{eval}$：给定同一公钥$\overline{pk}$加密下的两个密文$\bar{c}$、$\bar{c}'$，评估算法输出评估密文$\bar{c}_{eval}=f(\bar{c},\bar{c}')$。

$\mathrm{MKHE.Dec}\left((sk_1,\dots,sk_k),\bar{c}\right)\to m$：给定私钥份额$(sk_1,\dots,sk_k)$，和密文$\bar{c}$，解密算法输出$m$。

以 BGV 同态加密算法为例介绍它的 MKHE 扩展[82]，流程如下：

$\mathrm{MKBGV.Setup}\left(1^\lambda,1^\mathrm{L},1^\kappa\right)$：安全参数为$\lambda$，乘法深度为$L$，$\kappa$是密钥数量上界，运行$\mathrm{BGV.Setup}$获得公共参数$pp$，$a$是 CRS 参数。

$\mathrm{MKBGV.KeyGen}(pp)$：生成密钥对$(sk,pk)$，其中$sk=s$，$pk=(b=-as+te,a)$。

$\mathrm{MKBGV.Enc}\ (pk,m)$：给定公钥$pk$和消息$m$，加密得到密文$c=(c_0,c_1)\in R_q^2$，其中$c_0=rb+m+te_0$，$c_1=ra+te_1$。每个参与方都有一个索引$i$，设置有序的密文集合$S=(i)$。新的 BGV 密文为元组$\hat{c}=(c,S,L)$。

$\mathrm{MKBGV.Extend}((pk_1,\dots,pk_k),\hat{c})$：将 BGV 密文扩展到由$k$个参与方密钥加密的一个密文，将密文$c$设置成$k$个串联的子向量$\bar{c}=(c_1',\dots,c_k')\in R_{ql}^{2n}$。如果$i\in S$，令$c_i'=c_i$，否则$c_i'=0$。更新索引集合$S$。

$\mathrm{MKBGV.Eval}(\overline{pk}, f, \bar{c}, \bar{c}')$：给定$\overline{pk}$加密的两个密文$\bar{c}, \bar{c}'$，计算同态加法$\bar{c}_{add} = \bar{c} + \bar{c}'(mod\ q)$，计算乘法$\bar{c}_{mult} = \bar{c} \otimes \bar{c}'(mod\ q)$，然后使用评估密钥执行密钥置换降低密文维度，使用模数变换降低噪声。

$\mathrm{MKBGV.Dec}((sk_1, \ldots, sk_k), \bar{c}) \rightarrow m$：给定扩展密文$\bar{c}$和拼接私钥$\bar{s}$(在集合$S$中的参与方的私钥)，解密$(< \bar{c}, \bar{s} >)\ (mod\ q)(mod\ t) = (\sum_{i=1}^{k} < c_i', s_i > (mod\ q))\ (mod\ t)$。

MKHE 可以通过洋葱加密和解密技术将标准的 HE 扩展为用于恒定数目密钥的 MKHE，对明文重复以将密文扩展到其他密钥，解密是通过使用一组对应的密钥以加密时相反的顺序解密。单跳 MKHE 中，Clear 等人基于 GSW 提出了一种用于多身份同构 IBE 的编译器，也可以通过这种方式获得多密钥 HE。多跳 MKHE 中克服了早期 MKHE 的局限性，并允许在同态求值之后进行密文扩展。GSW、BGV、TFHE、BFV、CKKS 等都有对应的的多跳 MKHE 扩展[83][84][85]。同态加密打包技术是指将多个明文打包为单个密文，并以 SIMD（单指令多数据）方式对明文进行操作。CRS(Common Reference String）模型指的是所有参与者共享一个均匀分布的随机串（可信第三方提供），非交互场景下经常需要该模型。

3.2.4.3小结

表 14　不同 MKHE 架构的属性比较

方案	安全性	密文增长	协议轮数	多跳	密钥数量上限	CRS	打包	需要自举
LTV12[86]	NTRU	平方	-	√	√			
CM15[87]	LWE	平方	-			√		
MW16[88]		平方	2			√		
PS16[85]		平方	2	√	√	√		
BP16[89]		线性	2	√		√		√
DHRW16[90]	piO	平方	2	√				√
CZW17[82]	RLWE	线性	2	√	√	√	√	
YKHK18[91]		线性	2	√	√	√	√	
LZYH+19[92]		线性	2	√	√	√	√	
CDKS19[93]		线性	2	√	√	√	√	
AH19[94]		常数	4	√	√	√	√	
CCS19[95]	TLWE	线性	2	√	√	√		√

表 14 从安全性、密文增长方式、协议轮数、是否为多跳、是否有密钥数量上限、是否是 CRS 模型、是否支持打包以及是否需要自举对现有 MKHE 架构进行比较。现有方案的安全性假设有 NTRU、LWE、piO 和 RLWE。密文大小从早期工作中的平方增长改进为与最近工作中的密文维度呈线性增长。绝大多数方案的轮数都是 2 轮。除了 CM15、MW16 以外其它方案都是多跳的，这意味着在同态求值之后，可以将扩展密文进一步扩展到其他密钥。表中除 LTV12、DHRW16 以外的其它方案都要求密钥生成过程中使用 CRS 模型中的公共参数。基于 RLWE 的 MKHE 支持将多个明文打包在一个密文中，从而可以进行 SIMD 评估。CM15、MW16、BP16 和 DHRW16 没有密钥上限，但是前两个方案不支持多跳，后两个方案需要自举。

3.3 适用场景

MPC 方案可以通过本章介绍的各种技术方案进行实现，各类方案在实现性能、参与方数量等方面存在差异，在具体的应用中，可以根据实际的需求选用不同的实现方案。表 12 对混淆电路、秘密分享、同态加密三类 MPC 技术进行了对比，鉴于 OT 通常是与其它方案集成使用，在本节中未进行单独分析。

表 15　不同 MPC 技术对比

技术	参与方数量	计算轮数	开销	适用场景
混淆电路	通常为 2，BMR 等为 n	恒定轮数	通信开销大（和电路大小有关）	简单逻辑运算，小数据量；通常用来计算比较大小等非线性运算
秘密分享	一般≥3	线性轮数	计算和通信开销不断优化	可以计算线性运算和非线性运算，目前应用范围最广
同态加密	传统同态为 2，多密钥同态为 n	恒定轮数	计算开销大	目前主要用于同态加法和标量乘法，同态乘法以及同态比较开销大

GC-MPC 由于实现了底层的基本逻辑运算符，因此可面向不同的应用逻辑，具有通用性。混淆电路参与方一般为 2 方，BMR 等协议支持 n 个参与方。混淆电路的计算是恒定轮数的，计算开销主要是哈希函数的计算，计算开销较小；通信开销取决于电路门的数量。GC-MPC 在面对简单逻辑运算时有优势，如“比较”运算，但是在面对复杂运算时需要产生庞大的电路，导致通信量大，性能降低，因此不太适用于复杂计算场景，目前也没有出现针对大数据量复杂计算的处理平台，而在一些小数据量、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 相比 HE-MPC、GC-MPC，基础运算（特别是乘法）所需通信量较大，但是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，使之成为目前应用最广的 MPC 技术。秘密分享一般支持 3 或 4 个服务器，客户端的数量可以任意的。秘密分享的轮数不是恒定的，通常和要分享的秘密的数量成线性关系。秘密分享计算开销小具有高吞吐率，但是计算轮数取决于秘密分享的数量，具有高时延。秘密分享不仅可以计算加法和乘法这种线性运算，还可以计算比较大小，除法、截断等非线性运算。

FHE-MPC 计算一般具有如下特点：1）通常是逐位运算，例如 TFHE[67]（CGGI16）、FHEW[96]（CZW17）；或逐字运算，例如 BGV、BFV、CKKS。2）为防止 bootstrapping 问题产生很多额外计算代价。3)密钥和密文空间复杂度高，膨胀系数大。以上原因导致其计算性能相对很弱，特别是对于一些具有相对深度的布尔/算术电路，其计算效率难以被实际应用接受。

对逐字和逐位运算的全同态加密技术进行了比较，如表 16 所示。逐字运算同态加密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。逐位运算同态加密算法可以支持任意运算，但是性能显著低于逐字运算。阿里巴巴安全双子座实验室提出全同态密文技术“Pegasus”，可以高效桥接逐字和逐位运算两类全同态密文技术。

表 16　逐字和逐位运算全同态加密技术比较

	多项式运算	非多项式运算
逐字运算全同态加密（BFV/BGV/CKKS）	支持 packing，性能好	不支持，只能近似
逐位运算全同态加密（TFHE/FHEW）	不支持 packing，性能差	支持

在实际应用中，往往是多种运算的结合，将同态加密、秘密分享、混淆电路结合应用，来达到更好的性能。例如 GAZELLE 采用同态加密来计算加法乘法等线性操作，用混淆电路来计算比较大小等非线性操作，在同态加密和混淆电路两种技术切换的过程中用到秘密分享，来防止中间结果的泄露。

4. 应用及实现

4.1 应用场景

近年来，MPC 技术被广泛应用于机器学习、隐私集合求交、隐私查询等领域，此外还有一些常见的专用 MPC 方案，例如比较和排序[97][98]，保护私有信息几何计算[99][100]等。由于当前 MPC 在机器学习、隐私集合求交等领域研究更多，本节主要围绕这两项进行介绍。

4.1.1 机器学习

深度学习是机器学习的分支，是一种以人工神经网络为架构，对资料进行表征学习的算法，被广泛应用于计算机视觉、语音识别、自然语言处理等领域。卷积神经网络（Convolutional neural network, CNN）是深度学习的代表算法之一，由输入层、隐藏层和输出层组成，其中隐藏层由卷积层（Convolutional layer, CONV）、全连接层（Fully connected layers, FC）、池化层（Pooling layer）等组成。卷积层的计算是使用过滤器（Filter）对图像进行卷积遍历，即对过滤器矩阵和图像矩阵对应元素相乘然后求和。全连接层的计算是先乘权重矩阵再与偏移向量相加。池化层分为最大池化层（Max pooling）和平均池化层（Avg pooling），最大池化层的计算是取过滤器区域内所有神经元的最大值，平均池化层的计算是取过滤器区域内所有神经元的平均值。神经网络隐藏层和输出层都需要使用激活函数（Activation function），常用的激活函数有 ReLU、Sigmoid 和 Tanh 等，激活函数给神经元引入了非线性因素，使得神经网络可以任意逼近任何非线性函数，这样神经网络就可以应用到众多的非线性模型中。卷积层、全连接层和平均池化层中的运算为线性运算，最大池化层和激活函数中的运算为非线性运算。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，逻辑回归使用了 Sigmoid 激活函数。

基于 MPC 实现的隐私保护深度学习架构根据计算方数目可以分为 2PC 架构和多 PC 架构。2PC 架构中有 2 个计算方，在进行模型训练时各计算方通常是与数据拥有方独立的计算服务器，在进行模型推理时，通常其中 1 个计算方是数据拥有方本身。多 PC 架构中，计算方一般是 3 个或 4 个，数据拥有方的数量可以是任意的。

4.1.1.12PC 架构

本节对基于 2PC 架构的隐私保护深度学习进行介绍，这些方案通常由若干个 MPC 基础协议组合构建，包括 GC、OT、SS 和 HE。表 17 列出近年基于 2PC 架构的隐私保护深度学习，总结了其使用的 MPC 技术、运算类型、神经网络功能、准确率和性能等。

表 17　基于 2PC 架构的深度学习隐私保护

方案	MPC技术	支持非线性层	推理/训练	数据集	准确率（%）	推理性能
CryptoNets(2016)[101]	HE	否	推理	MNIST	99	59000 次推理/hour
SecureML(2017)[102]	SS, GC, HE	否	训练推理	MNIST	93.4	运行时间：4.88s
MiniONN(2017)[103]	AHE, GC, SS	是	推理	MNIST等	99	运行时间：9.32s 通信开销：657.5MB
EzPC(2017)[104]	GC, SS	是	推理	MNIST等	99.2	运行时间：5.1s 通信开销：501MB
DeepSecure(2018)[105]	GC	是	推理	MNIST	/	运行时间、通信开销优于 CryptoNets 527 倍、1000 倍
Chameleon(2018)[106]	GC, GMW, SS	是	推理	MNIST等	99	运行时间优于 MiniONN 4.2 倍
GAZELLE(2018)[107]	AHE,GC，SS	是	推理	MNIST等	/	运行时间优于 Chameleon 20 倍
XONN(2019)[108]	GC, SS	是	训练推理	MNIST等	99	运行时间优于 GAZELLE 3.4 倍
QUOTIENT(2019)[109]	COT, SS	是	训练推理	Thyroid 等	79.5-99.38	运行时间优于 SecureML 13 倍
Delphi(2020)[110]	HE, SS, GC	是	推理	CIFAR-100 等	/	运行时间、通信开销优于 GAZELLE 22 倍、9 倍
CrypTFlow2(2020)[111]	SS, OT, HE	是	推理	ImageNet	79.9-91.9	运行时间、通信开销优于 Delphi 22 倍、9.3 倍

以上方案中，CryptoNets、SecureML 不支持神经网络中的非线性层运算，其它方案都支持非线性层运算。SecureML、XONN 和 QUOTIENT 同时支持神经网络的训练和推理，其它方案只支持神经网络推理。早期的隐私保护机器学习[101]

使用层数较少的神经网络以及单一的数据集 MNIST 和 CIFAR-10，而 QUOTIENT 、Delphi 和 CrypTFlow2 使用更复杂的神经网络模型 ResNet、SqueezeNet、DenseNet121，以及大规模数据集 CIFAR-100、ImageNet 等。由于模型和数据集不同，部分方案之间的准确率没有可比性。

使用 2PC 方案进行模型推理时满足 CS 模型，2 个计算方分别为客户端（私有数据拥有方）和服务器（模型参数拥有方），在推理计算过程中保证数据和模型不能被对方获得。使用 2PC 方案进行模型训练时，2 个计算方是不合谋的服务器，客户端可以是任意数量的数据拥有方。

1) 神经网络推理

CryptoNets 是 Dowlin 等人提出的使用全同态加密（FHE）来实现隐私保护深度学习，可以对同态加密数据进行神经网络推理，只支持线性层的运算，不支持非线性层的运算。该方案的推理准确率为 99%，每小时可完成 59000 次推理。

MiniONN 是 Liu 等人提出的基于遗忘神经网络(oblivious neural network，简称 ONN）的两方安全计算框架。使用 SS、HE 和 GC 来完成神经网络的推理，客户端对私有数据进行秘密拆分，数据份额分别由客户端和服务器持有，服务器对全连接层的权重参数计算同态密文并发给客户端，客户端对权重密文进行同态运算得到权重和输入乘积的密文，然后客户端和服务器进行信息交互分别得到全连接层输出的份额。该方案中可以使用 GC 来实现 ReLU 激活函数，并且构造了非线性激活函数 Sigmoid 的多项式近似函数。该方案训练模型的准确率为 99%，运行时间为 9.32s，通信开销为 657.5MB。

EzPC 是 Microsoft 实现的安全两方计算编译器。此前的编译器例如 Wysteria 等都只能实现算术电路或布尔电路，EzPC 是第一个将算术和布尔电路结合起来的编译器，该编译器的后端实现了 ABY 协议。EzPC 编译器将源程序作为输入，输出为 C++程序，将输入程序分为公共组件和秘密组件：公共组件转换为常规 C++代码；秘密组件转换为对 ABY 的调用。EzPC 运行时间为 5.1s，通信开销为 501MB，比先前的工作快 19 倍。

DeepSecure 是 Rouhani 等人提出的基于 GC 的神经网络推理框架，方案对 GC 组件进行优化并引入了降低开销的预处理技术。DeepSecure 运行时间比 CryptoNets 快 527 倍，通信量多 1000 倍。

Chameleon 是 Riazi 等人提出的混合安全计算框架，该框架用加性 SS 执行线性操作，利用 GMW 或 GC 协议执行非线性操作。Chameleon 需要一个半诚实第三方在离线阶段生成相关随机数，在线阶段两个参与方结合相关随机数和私有输入进行协同计算。Chameleon 模型准确率为 99%，运行时间比 MiniONN 快 4.2 倍。

GAZELLE 是 Juvekar 等人提出的基于 AHE 和 GC 的安全神经网络推理方案。利用 AHE 来执行线性运算，使用 GC 来执行非线性运算，并且用加性秘密分享来实现 AHE 和 GC 之间的转换。GAZELLE 的在线阶段运行时间比 MiniONN 和 Chameleon 快 20 - 30 倍。

Delphi 是 Mishra 等人提出的 GAZELLE 的优化方案，降低了在线阶段的计算和通信开销。该方案使用 HE 来计算线性层，使用 GC 或 SS 来计算非线性层。在预处理阶段客户端和服务器通过 HE 生成模型权重等参数的份额、生成混淆电路、生成 Beaver 三元组；在线阶段直接在预处理阶段生成的数据份额上执行运算。对于非线性层，可以使用 GC 来精确计算，或者使用 Beaver 三元组来计算非线性层的线性近似函数。Delphi 在线阶段协议运行时间比 GAZELLE 快 22 倍，通信量少 9 倍。

CrypTFlow2 是 Microsoft 提出的安全两方深度神经网络推理方案。该方案使用 SS 和 OT 实现了比较大小运算，进一步也解决了 ReLU、Max pooling 和 Avg pooling 等非线性层的安全计算。CrypTFlow2 支持两种不同的线性层：基于同态 SEAL 和 Delphi 实现的线性层、基于 OT EMP-toolkit 实现的线性层，通信效率比此前的方案提高了 20-30 倍。

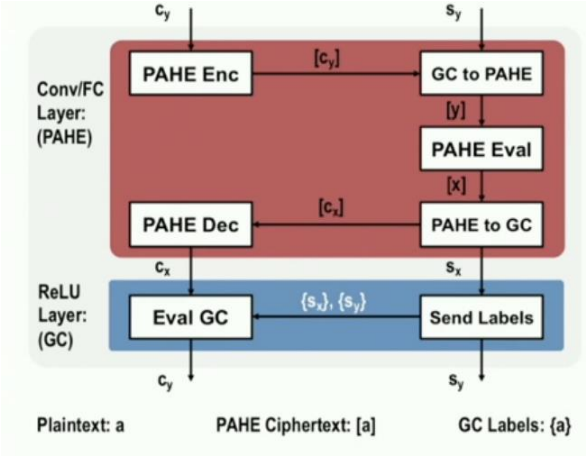

图 19　Gazelle 框架

以 GAZELLE 为例详细介绍 MPC 在深度学习中的应用，该方案中客户端和服务器使用 AHE 执行线性运算，使用 GC 执行非线性运算。评估线性层时如图 19 所示，在 AHE 阶段开始时，客户端和服务器拥有y（客户端私有输入）的加法份额c_y、s_y，初始阶段$(c_y, s_y) = (y, 0)$。首先，客户端对自己的份额进行同态加密并发送给服务器，服务器也对自己份额进行同态加密并将加密后的密文与客户端的份额密文相加获得$c_y + s_y = y$的密文$[y]$。然后服务器使用同态运算来计算线性层，服务器此时拥有客户端的图像加密输入$[y]$以及自己原本拥有的神经网络参数，服务器对 $[y]$和神经网络参数计算标量乘法，得到线性层的输出密文$[x]$。评估非线性激活层时，服务器持有密文$[x]$，客户端持有私钥，服务器将$[x]$拆分为两个加法份额，具体如下：服务器选取随机数r并计算$[x] + [r] = [x + r]$发送给客户端，客户端解密获得份额$c_x = x + r$，服务器设置自己的份额为$s_x = r$。c_x和s_x作为线性层的输出，同时也是非线性层的输入，使用 GC 来评估非线性激活函数。

2)　神经网络训练

SecureML、QUOTIENT 和 XONN 支持神经网络训练，同时也支持神经网络的推理。

SecureML 是 Mohassel 等人提出的安全两方神经网络训练和推理框架。在训练过程中，离线阶段利用 HE 或者 OT 为在线阶段生成 Beaver 三元组；在线阶段利用秘密分享计算加法和乘法运算，使用混淆电路计算激活函数。由于使用混淆电路计算激活函数 ReLU 的开销大，将激活函数替换为平方函数，用 MNIST 数据集在 LAN 下来自定义网络进行训练和推理，只获得了 93.4%的准确率。

QUOTIENT 是 Agrawal 提出的安全两方神经网络训练和推理框架。该模型中有两个不合谋的服务器，在神经网络的训练过程中将权重分层为{-1,0,1}，提出了基于 COT 的三元矩阵矢量乘法计算协议，结合布尔秘密分享和加性秘密分享来提高效率。此前 SecureML 只完成了 MNIST 数据集在全连接神经网络上训练的结果，而 QUOTIENT 实现了对 5 个数据集在神经网络 ResNet-20 上的训练，获得了

50 倍效率的提升，准确率提高 6%。QUOTIENT 与明文训练相比，准确率只降低了 0.1%-2.17%。

XONN 是 Riazi 提出的安全二进制神经网络训练和推理框架。与 SecureML、QUOTIENT 系统架构不同，XONN 将神经网络参数和激活单元都限制为二进制形式（+1 或-1），该方案在准确性和通信开销之间进行折中。XONN 将神经网络中的线性运算分为两种：一种是整数和二进制数的向量乘法，对于神经网络的第一层，客户端的输入不是二进制的，而服务器拥有的神经网络权重是二进制的，使用 OT 来完成这种线性运算；另一种是两个向量都是二进制的乘法，使用 XNOR 运算所取代，而 XNOR 运算在 Free-XOR 下是不需要传输密文的。XONN 的表现优于 GAZELLE 7 倍。

4.1.1.2多 PC 架构

本节对基于多 PC 架构的隐私保护深度学习方案进行介绍，多 PC 方案通常具有 3 个或 4 个计算方和任意数量的客户端，包括ABY³、PrivPy、SecureNN、TRIDENT 和 CrypTFlow 等，如表 18 所示。

表 18　基于多 PC（3/4PC）架构的深度学习隐私保护

框架	关键技术	门限方案	推理/训练	数据集	准确性(%)	性能
ABY³(2018)[112]	SS, GC	是	训练，推理	MNIST	93-99	训练比 SecureML 快 80-55000 倍
SecureNN(2018)	SS	否	训练，推理	MNIST	99	比 SecureML 快 8-407 倍
PrivPy(2018)	SS	是	训练，推理	MNIST	/	优于ABY³
TRIDENT(2019)[113]	SS,GC	是	训练，推理	MNIST 等	93-98.3	比ABY³快 158 倍
CrypTFlow(2020)[114]	SS,GC	否	推理	ImageNet	与明文模型相同	半诚实模型：30s 恶意模型：2min

ABY³、PrivPy 和 TRIDENT 实现了门限方案，能够容忍计算节点的单点失效。CrypTFlow 目前只支持神经网络推理，其它几个方案支持神经网络训练和推理。表中还列出了准确率和性能，此外，这些方案都支持非线性层运算和批处理计算。

ABY³(Arithmetic-Binary-Yao) 是 Mohassel 等人提出的 3PC 方案，满足诚实方占多数的情况。该方案包含三种数据类型：算术秘密分享、二进制秘密分享、Yao's GC，根据运算的特点在三种数据中切换：当计算加法和乘法时使用算术秘密分享，计算比较大小等非线性运算时使用二进制布尔秘密分享和 Yao's GC。ABY³将上述底层的安全计算方案应用于机器学习的线性回归、逻辑回归、神经网络和卷积神经网络的训练和推理中，ABY³对 MNIST 数据集进行训练，得到的模型准确性范围为 93%（线性回归）到 99%（CNN），神经网络和线性回归的训练阶段运行时间分别比 SecureML 快 80-55000 倍，神经网络的推理运行时间比 MniONN、Chameleon 快 932.9、270 倍，通信量少 126.4、2.5 倍。

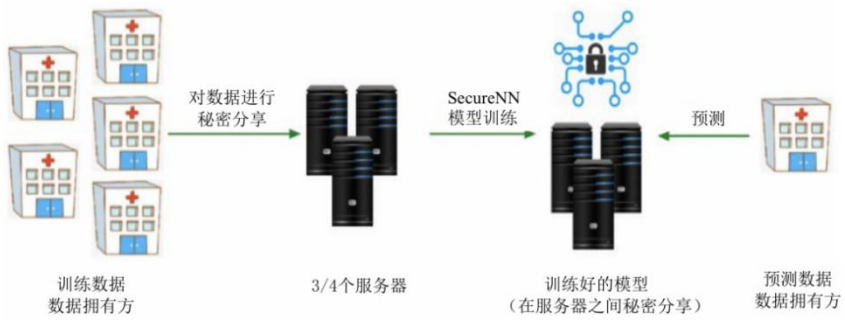

图 20　SecureNN 框架

SecureNN 是 Wagh 等人提出的 3PC/4PC 方案，架构如图 20 所示。数据拥有方可以是任意数量，服务器可以是 3 个或 4 个，所有服务器的关系并不是对等的，3 方模型中有 2 个服务器负责计算，1 个服务器辅助计算。该方案基于秘密分享，实现了加法、矩阵乘法、比较大小、计算最高有效位，除法等多方安全计算协议，并且基于这些安全协议，实现了深度神经网络中的卷积、全连接层等线性层，以及 ReLU、Max pooling 等非线性层。SecureNN 对 MNIST 数据集进行模型训练和推理，得到的模型的准确率均高于 99%，SecureNN 的模型训练时间比 SecureML 等方案提高了 8-407 倍，模型推理时间比 MiniONN、Chameleon、GAZELLE 分别快 42.4 倍、27 倍、3.68 倍。

PrivPy 是 Li 等人提出的 2-out-of-4 方案，客户端将秘密分享的数据发送给服务器，然后服务器基于这些共享执行隐私保护计算，四个服务器中任意两个服务器可以恢复出计算结果。PrivPy 由两部分组成：语言前端和计算引擎后端。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 支持三种后端：PrivPy 设计的后端（2-out-of-4 秘密分享协议）、SPDZ 和$\mathbf{ABY^3}$。在训练和推理方面，PrivPy 的性能均优于$\mathbf{ABY^3}$，WAN 设置比 LAN 设置中优势更明显。

TRIDENT 是 Chaudhari 等人提出的 4PC 协议，可以容忍一个恶意参与方。协议中实现了三种技术的混合框架，在算术秘密分享、布尔秘密分享和混淆电路之间进行切换。使用布尔分享来完成比较大小和比特提取，使用算术秘密分享来计算加法和乘法，使用混淆电路来计算环上的除法。TRIDENT 使用 Boston、Weather 和 CalCOFI 数据集来训练线性回归模型，使用 Candy、Epileptic 和 Recipes 数据集来训练逻辑回归模型，使用 MNIST 来训练 CNN，得到模型的准确率范围是 93%（线性回归）到 98.3%（CNN），在训练和推理阶段的运行时间分别比$\mathbf{ABY^3}$提高 187、158 倍。

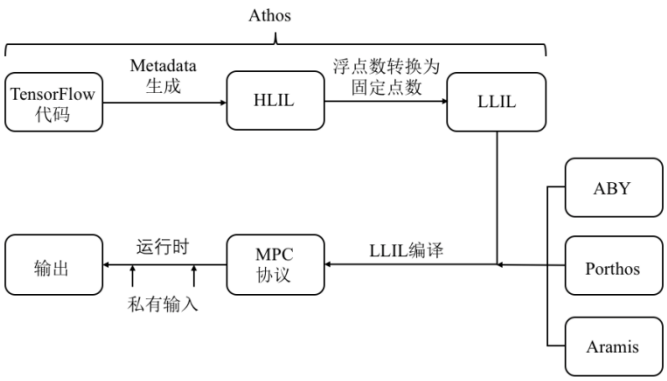

图 21　CrypTFlow 架构

CrypTFlow 是 Microsoft 提出的将 TensorFlow 推理代码转换为 MPC 协议的方案。CrypTFlow 构建了 Athos、Porthos 和 Aramis 三个组件，如图 21 所示。Athos 是一个编译器，主要实现了三个功能：将 TensorFlow 代码编译为高级中间语言(HLIL)；自动执行转换将浮点数转换为定点数(深度学习用浮点数来计算，MPC 使用定点数计算）；使用低级中间语言（LLIL）将浮点 HLIL 代码编译为定点代码，LLIL 可以衔接不同的加密后端（ABY、Porthos、Arimis）。Porthos 是在 SecureNN 基础上改进的三方协议，以提高通信效率。SecureNN 中将两个矩阵的卷积计算转换成两个变形矩阵的乘法。然而变形矩阵有一些冗余的参数会增加通信开销，在 Porthos 中，参与方先对原始矩阵和 Beaver 元组进行计算然后交换份额，参与方在本地对交换后的矩阵变形，最后再对变形矩阵计算矩阵乘法。对于非线性层的改进是通过修改“计算最高有效位”和“份额转换协议”的使用方式来降低通信开销。Aramis 使用 SGX 技术将 MPC 协议从半诚实安全模型转换为恶意安全模型。该系统与明文 TensorFlow 的推理准确性相近，半诚实安全模型的运行时间约为 30 秒，恶意安全模型的运行时间不到 2 分钟。

4.1.2 隐私集合求交

隐私集合求交（Private Set Intersection,PSI）计算属于多方安全计算领域的特定应用问题，不仅具有重要的理论意义，也具有很强的应用价值。

PSI 中一般有两个参与方，每个参与方拥有一个集合分别记作 X 和 Y。双方想计算两个集合的交集，但不想泄露其它额外的信息(对方集合中非交集的元素)。现实生活中很多数据都可以用集合来表示，并用集合间的隐私保护计算来完成一些数据计算问题，因此 PSI 计算有很广泛的应用场景：隐私保护相似文档检测、私有联系人发现等。

PSI 可以分类为基于 Hash 函数的 PSI、基于公钥加密的 PSI、基于混淆电路的 PSI、基于 OT 的 PSI 和基于 FHE 的 PSI。

1) 基于 Hash、基于公钥加密体制的 PSI 协议（基于 Hash 的 PSI 协议存在一定安全问题）

最基本的 PSI 协议是协议双方对自己的集合计算哈希值，其中一方把自己的哈希值发送给另一方，另一方计算交集。该协议非常有效，但是当输入取值范围很小时存在安全问题：恶意方计算取值范围内所有元素的哈希值，然后进行交集运算可以得到另一方的所有输入。

基于公钥加密体制的方法主要对集合元素进行复杂的公钥加密操作，并在密文上进行计算。主要有基于 Diffie-Hellman 密钥交换的 PSI 协议和基于 RSA 盲签名的 PSI 协议。其中，基于 Diffie-Hellman 密钥交换的 PSI 协议，可以结合 SM2 密钥交换协议来实施。

2) 基于 GC 的 PSI 协议

基于 GC 的 PSI 协议主要思想有两种：将元素映射到二进制向量，通过向量的与运算求集合交集；通过电路解决隐私保护的元素相等性判断问题，然后通过集合中元素的两两比较计算集合交集。

Huang 等人提出了 3 种基于混淆电路的 PSI 协议的实现[115]，分别为 BWA(Bitwise-AND)、PWC(Pairwise-Comparisons)、SCS(Sort-Compare-Shuffle)，分别适用于不同的集合规模。BWA 协议通过 GC 对参与方向量进行位与运算得出交集集合。PWC 协议通过混淆电路首先对两方元素进行 XOR 运算再进行 OR 运算，来判断两方元素是否相等。SCS 协议中双方先在本地对各自集合元素排序，再通

过 GC 按序合并，对合并后的相邻元素进行相等性判断。Pinkas 等人提出通过哈希表的方法对 Huang 的方案进行改进[116]，Ciampi 等人提出的 PSI 协议[117]比 Pinkas 的协议具有更好的性能。

3）基于 OT 的 PSI 协议

基于不经意传输的 PSI 协议主要通过 OT 协议进行向量或元素相等性判断来计算集合交集。Dong 等人于 2013 年提出了基于布隆过滤器、秘密分享、OT 扩展协议的 PSI 协议[118]，该协议是第一个可处理元素达到亿级别大小的 PSI 协议。

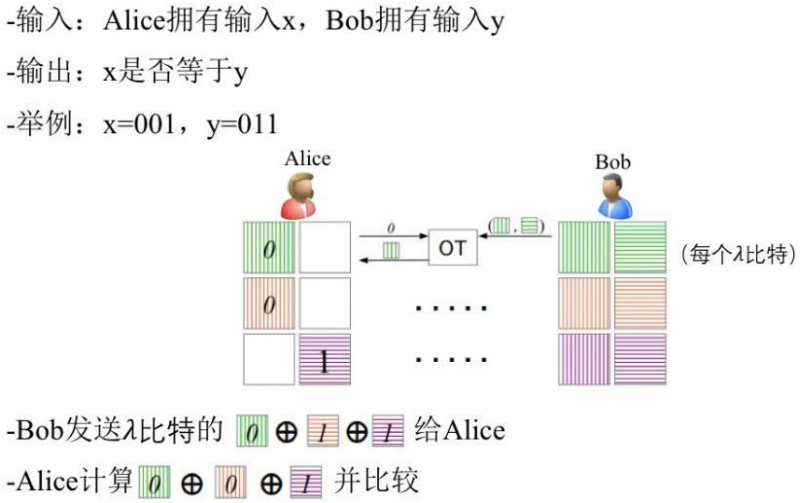

图 22 基于 OT 的 PSI 协议

Pinkas 等人在 2015 年提出了一个 PSI 协议[119]，通过 OT 扩展实现隐私相等检测，结合高效的哈希技术来提高协议效率，如图 22。协议中有两个参与方 Alice 和 Bob，分别拥有两个集合，假设 Alice 的输入是$x = 001$，Bob 的输入是$y = 011$，协议的输出是x是否等于y。Bob 分别为 0 和 1 随机采样两个λ比特长的字符串，然后 Bob 和 Alice 执行 OT 协议，Alice 依次获得第一个输入比特 0、第二个输入比特 0、第三个输入比特 1 所对应的字符串，并对这三个字符串求异或。Bob 的输入比特是 011，他对自己的输入比特 0、1、1 所对应的三个字符串求异或，将结果发送给 Alice。Alice 对比自己计算的异或值和 Bob 计算的异或值，来判断双方输入是否相等。Vladimir Kolesnikov 等人在 2016 年对 Pinkas 等人提出的协议进行改进[120]，综合使用了 OT、OT 扩展、布谷鸟哈希和不经意伪随机数生成器，该协议在用于生成大量 OPRF 实例时特别有效。Rindal 等人于 2016 年提出了恶意安全模型下的 PSI 协议[121]，采用 Cut-and-Choose 技术来抵御恶意客户端的攻击。Melissa 等人于 2020 年提出的协议[122]在通信和计算开销之间有了一个很好的平衡。

4）基于 FHE 的 PSI 协议

基于 FHE 的 PSI 协议，是由拥有较小集合的一方将数据加密后发给另一方，另一方进行全同态计算后将结果发给对方解密。Hao Chen 等人构造了一种高效的基于全同态的 PSI 协议[123]，该协议拥有很小的通信代价。

4.1.3 隐私信息检索

隐私信息检索（Private Information Retrieval, PIR）技术是指用户向数据库进行检索时，使用户的隐私信息（即用户感兴趣的信息）不被泄露的情况下完成检索的技术。这一技术在诸如患者信息查询、域名申请和专利申请等方面有广大的应用场景,同时在信息学理论上和诸如不经意传输、秘密分享、同态加密、LDC（locally decodable codes）等问题紧密相连。从 1995 年 Benny Chor 等人开展对这一问题的研究以来[124]，PIR 的探索发展主要分为两个领域：信息论安全隐私信息检索（IT-PIR）和计算安全隐私信息检索（CPIR）。

1） 信息论安全隐私信息检索（IT-PIR）

在用户检索时，将整个数据库的内容发送给用户，这种方法虽然十分低效，但是却是在单数据库条件下实现信息论安全的唯一一种 PIR 方案。对于更一般的多数据库情况，记数据库的个数为 N，每个数据库的条目数为 K，有通信开销更低的方案。

较早的研究中一般假设数据库中的每条信息均为 1 比特，并研究在这一条件下 PIR 方案的整体通信复杂度，即考虑用户上传至数据库的内容和用户从数据库下载的内容的总量。Chor 等人的研究基于秘密分享技术，提出了使用多项式插值法和覆盖码法（Covering Code）实现通信复杂度为$O(K^{\frac{1}{N}})$的 PIR 方案。近些年的研究则考虑数据库中的每条信息长度为任意值的情况。在这种情况下，用户上传的开销相比下载开销可以忽略。研究者一般用下载开销 C 的倒数，即用户从下载的每比特中获取的关心信息的比特数评价 PIR 方案的效率。

在 Hua Sun 等人的研究中[125]，作者证明了 PIR 的效率上界为$C = (1 + 1/N + 1/N^2 + \cdots + 1/N^{K-1})^{-1}$，并给出了相应的 PIR 方案。该方案中发往不同数据库的查询对称、每条查询关于条目对称，保证了方案的隐私性；利用下载信息中用户不关心的部分查询新的关心的信息，达成了理想的效率。下面举例对该方案进行简要介绍：

假设有$N = 2$个数据库 DB1 和 DB2，每个数据库中有$K = 2$个条目，分别是W_1和W_2，每个条目的信息长度为 4。记W_1中四位比特按顺序为$[a1, a2, a3, a4]$，W_2中四位比特按顺序为$[b1, b2, b3, b4]$。

a） 假设用户希望查询信息W_1，那么先在 DB1 的查询序列中写下$a1$，向 DB1 询问W_1的第一位的值。由数据库间对称性，在 DB2 的查询序列中写下$a2$，向 DB2 询问W_1的第二位的值。

b） 再在每条查询中分别写下$b1$和$b2$，通过查询序列关于条目的对称性保证隐私性。

c） 现在我们获取了不关心的数据$b2$，可以利用它在数据库 1 中查询$a3 + b2$，以获取信息$a3$的同时不泄露隐私。

d） 最后由数据库间对称性，在 DB2 中查询$a4 + b1$以获取$a4$。

查询序列的编写过程如下表：

表 19 查询序列编写过程

a)	DB1	DB2	b)	DB1	DB2	c)	DB1	DB2	d)	DB1	DB2
	$a1$	$a2$		$a1, b1$	$a2, b2$		$a1, b1$ $a3 + b2$	$a2, b2$		$a1, b1$ $a3 + b2$	$a2, b2$ $a4 + b1$

假设用户希望查询信息W_2，类似地，查询序列为

表 20　查询序列

DB1	DB2
$a1, b1$	$a2, b2$
$a2 + b3$	$a1 + b4$

这样用户就可以通过简单计算得知W_1和W_2的值，从而验证了这个方案的正确性。为了达成信息论安全，使得每种可能的查询序列组合在数据库看来概率均等，用户在实际询问前需要再执行一轮置乱操作：置乱询问的$[a1, a2, a3, a4]$和$[b1, b2, b3, b4]$和数据库中对应条目的比特位置的关系，这种置乱应当是均匀且独立同分布的。由于每个数据库的查询序列中只包括每个条目中随机选取的一个比特和从两个条目中各随机选取的一个比特的和，而随机排列是用户生成且均匀的，所以每种查询序列的可能性是均等的，从而保证了隐私性。

IT-PIR 也关心多数据库条件下数据库间串通，或者有数据库不响应查询的问题[126]：假设 N 个数据库中有 T 个数据库互相串通，记为 T-PIR 问题。T-PIR 的效率上界为$C = (1 + T/N + T^2/N^2 + \cdots + T^{K-1}/N^{K-1})^{-1}$，若 M 个数据库中仅有 N 个数据库响应查询，则是健壮 T-PIR 问题，该问题的效率上界为$C = (1 + T/N + T^2/N^2 + \cdots + T^{K-1}/N^{K-1})^{-1}$。注意到该 PIR 方案的效率和不需要考虑健壮性的情况一致，但随 M 的增加，方案的总通信复杂度会增加以应对不确定性。

2)　计算安全隐私信息检索（CPIR）

对于现实的应用场景而言，由于数据库的计算能力有限，可以使用密码学、编码技术等方式实现单数据库条件下的计算安全隐私信息检索。早期的研究基于二次剩余问题[127]和Φ假设（Φ Assumption）[128]等提出了经典方案。近年来随着新的非对称加密方法提出有一些新方案涌现，但是由于非对称加密所依赖的数学难题扩展性较强，实际应用的方案迭代比较缓慢。从实践的角度而言，CPIR 有三个不同的优化方向[129]：用户主导的方案，通过增加数据库返回的信息降低用户查询信息长度；数据库主导的方案，通过增加用户查询的长度降低返回信息的长度；均衡方案，注重优化整体的通信复杂度。

除了基于非对称加密的 CPIR 方案之外，也有研究提出了基于全同态加密的 CPIR 方案.早期的研究一般以最优化运算成本或通信复杂度展开。Gentry 的研究[130]提出了以通信复杂度$O(\gamma \log n)$实现数据库中的信息均为 1 比特假设下的 PIR 方案，但该方案的运算成本较高。Xun Yi 等人的研究[131]更进一步，实现了一般情况下的基于全同态加密的 CPIR，在保持通信量最优化的情况下降低了用户端的运算需求，但数据库端的运算量仍嫌偏大。随后提出的 SealPIR 则注重于降低运算成本，提高了数据库处理大量 PIR 请求的能力。这一方向最新的进展由 Asra Ali 等人提出，改进了 SealPIR 和 Gentry 的成果，使用 MulPIR 在运算量和通信成本之间寻求平衡的同时也大幅优化了追求最优通信量或最低运算量情况下的表现。相比之前的研究，在最优化通信量的情况下降低了 60%的计算成本，在最优化通信量的情况下降低了 85%的计算成本。该研究还提出了针对数据库特征进行优化的方案：在条目数量多，条目长度短的数据库中利用用户辅助的 Gentry 算法可以取得更好的整体效率；在条目数量较少，单个条目长的数据库中，MulPIR 则有着显著更优的效率。

基于上面介绍的不同类型的 PIR，结合现实需求，有更多的 PIR 方案被提出和开发。如除了保护用户隐私外也保护数据库隐私的对称 PIR（SPIR），不同数据库储存的数据不一致条件下的 PIR，数据库之间非同步检索的 PIR，在分布式储存条件下 MDS 编码数据库的 PIR 等。

4.2 MPC 产品与系统

基于成熟的理论研究成果，MPC 产业化应用不断发展，国内外存在很多从事 MPC 相关的技术和产品研发的企业。从技术应用方向看，MPC 系统可以分类为通用计算平台和隐私保护机器学习系统。

4.2.1 通用计算平台

1) Sharemind MPC

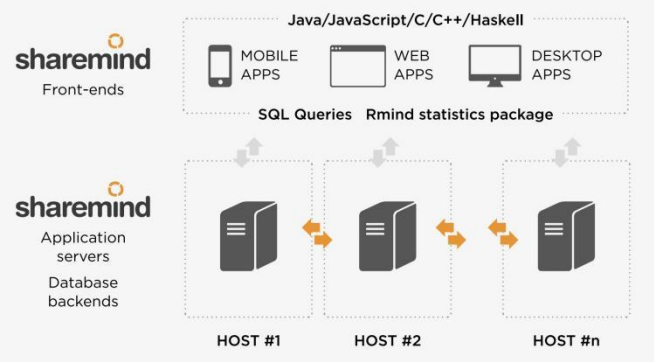

图 23 Sharemind 框架

Sharemind MPC 是由 Cybernetica 公司推出的具有通用编程功能的多方计算产品。该产品前端支持任意数目异构客户端，服务端配备多个应用服务器和数据库，其应用框架如图 23 所示。

Sharemind MPC 客户端提供数据输入、数据格式转换和数据分析功能；服务端作为 MPC 计算节点，集安全数据存储、逻辑执行单元、堆栈缓存于一体，搭载加密计算引擎，实现隐私保护的数据计算程序。

Sharemind MPC 服务端程序使用 SecreC 语言编写，客户端使用 Java、C 等常用语言编写。Sharemind MPC 允许用户通过客户端 API 使用基于 Sharemind 的工具包，包括库、头文件和开发文档等，用于自定义开发平台应用以及将 MPC 技术与现有系统集成，客户端 API 支持 C、C++、Java、Haskell、JavaScript 等语言的应用程序调用。

2) PrivPy MPC

PrivPy 是由华控清交推出的多方安全计算平台，是针对大数据应用系统和人工智能系统而开发的数据安全融合计算产品，对外提供通用的安全计算能力。该计算平台主要由 MPC 数据服务模块、MPC 计算引擎模块、MPC 调度及管理模块组成，并对上提供各类算法服务，系统框架如图 24 所示。MPC 计算引擎基于 PrivPy 技术框架实现，采用 4 个独立的计算节点完成计算，对外提供多方数据的加法、乘法、比较等基础函数库功能。算法服务模块中包含基础函数库、应用算法库和 MPC 算法 IDE 模块，MPC 算法 IDE 模块支持在线编写 Python 语言和 Python 语言对应的函数库和算法库。

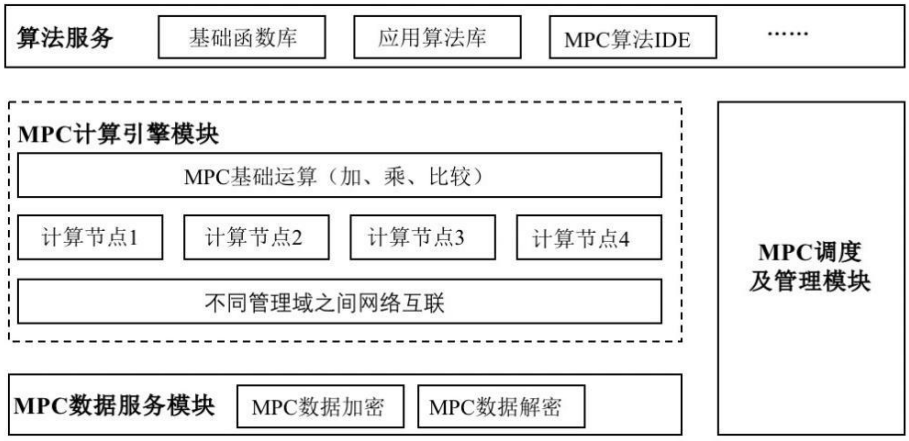

图 24　PrivPy MPC 通用计算平台框架图

PrivPy MPC 为上层应用提供了通用的计算模型，具体特点如下：

a) 提供了高级语言 python 编程接口，将用户自主定义的算法逻辑转化为基本的运算单元并输入给计算节点进行计算。

b) 支持统计分析、线性代数运算、机器学习等多种算法类型并提供了算法库，因此可面向多类应用服务进行扩展。

c) 支持多种数据类型，包括整数、实数等。同时，为支撑实际应用中对实数运算精度的要求，PrivPy MPC 使用了一种离散方法将实数映射到整数域。

d) 支持对底层协议的扩展。在底层协议上，除了 PrivPy 之外，计算引擎可以扩展，支持添加其他协议，如 SPDZ。

3) JUGO

JUGO 是由矩阵元推出的多方安全计算平台，提供半诚实通用两方计算能力，该平台主要由 MPC-IDE、MPC-SDK 等模块构成，支持 BMR 等 GC 协议和同态加密算法/协议，系统框架如图 25 所示。

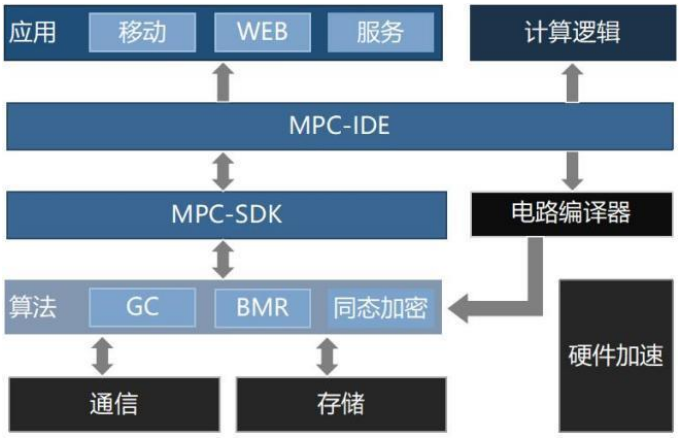

图 25　JUGO 多方安全计算平台框架图

MPC-IDE 实现计算逻辑的编写。开发者使用指定的 Frutta 语言编写算法，通过 MPC-IDE 模块编译成可执行的电路文件。Frutta 语言是矩阵元为多方安全计算的算法电路文件生成而专门定制的编程语言，目前只支持指定的 IDE 进行编译运行。

MPC-SDK 为多方安全计算应用所提供的开发工具包，集成了 MPC 算法，作为数据执行方直接为计算逻辑提供者服务。MPC-SDK 提供了丰富的应用开发接口，而且还是 MPC 的计算节点，通过对 SDK 的调用，应用可以实现数据上传、算法调用、计算邀请、任务受理、MPC 本地计算等功能，使得数据在不离开本地的情况下，获取计算结果。MPC-SDK 内部算法实现 GPU、FPGA 等硬件加速，使协同计算过程更快地完成。

4) Morse MPC

图 26 蚂蚁链摩斯平台架构

蚂蚁链摩斯（Morse MPC）是由蚂蚁集团推出多方安全计算平台，提供秘密分享、混淆电路、同态加密等多种多方安全计算技术的实现和应用，解决数据协同计算过程中的数据安全和隐私问题。该平台支持文本文件、云存储 OSS、关系行数据库 MySQL 及列数据库 HBase 等数据源。该平台架构自下而上如图 26 所示，底层算法组件和安全引擎向上提供安全计算能力。中间安全计算产品层封装了平台核心功能，为顶层不同解决方案提供安全模型、安全统计和安全服务。

4.2.2 隐私保护机器学习系统

1) FATE

FATE (Federated AI Technology Enabler) 是微众银行 AI 部门发起的开源项目，近年来一直在不断更新，为联合学习生态系统提供了可靠的安全计算框架。FATE 项目使用多方安全计算 （MPC） 以及同态加密 （HE）技术构建底层安全计算协议，以此支持不同种类的机器学习的安全计算，包括逻辑回归、基于树的算法、深度学习和迁移学习等。FATE 提供了模块化的、可扩展的建模管道，方便开发人员进行数据使用、联合训练和模型管理等，但是由于 FATE 提供的是算法级接口，所以实现自定义的联合学习算法只能修改其源码。

2) EzPC

Microsoft 的 EzPC 项目于 2020 年开发了一个开源系统 CrypTFlow，该系统将 TensorFlow 推理代码作为输入，并自动将其编译为针对同一代码的高效安全计算协议，能够在不泄露数据隐私的前提下进行联合学习的计算。CrypTFlow 系统主要拥有以下组件：Athos 作为端到端编译器可将 TensorFlow 编译为各种半诚实的 MPC 协议；Porthos 是经过改进的半诚实的三方协议（SecureNN），可为系统提高通信效率；Aramis 能将半诚实行为假设的 MPC 协议转换为恶意行为假设的 MPC 协议。这些组件可以独立使用，但组合为 CrypTFlow 系统可以更高效地构建 MPC 协议。

3) TF Encrypted

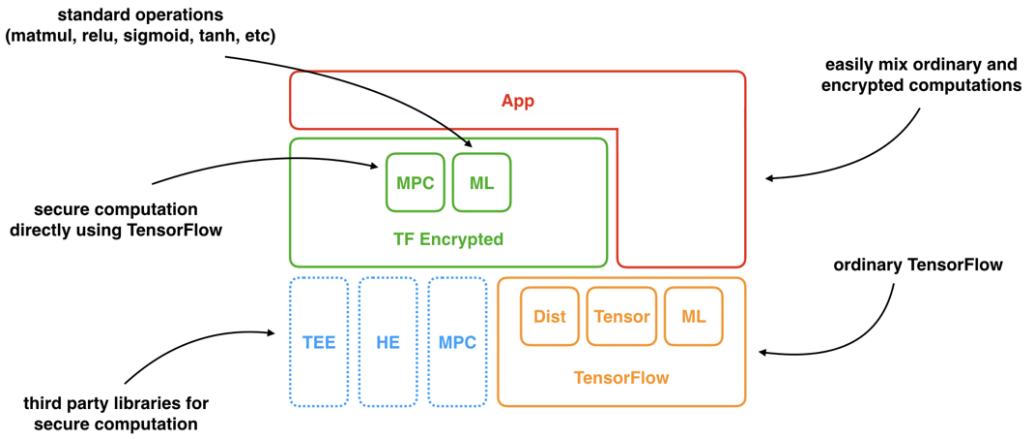

图 27 TF Encrypted 应用的架构

TF Encrypted 是一个开源社区项目，于 2018 年由 Dropout Labs（Cape Privacy 公司前身）首次提出并发布，目前主要用于实验及研究。TF Encrypted 是一个建立在 TensorFlow 上的加密机器学习框架。它抽象了大多数的底层复杂性，提供了高级界面，目的是让没有专业背景知识的用户也可以方便地使用隐私保护机器学习。如图 27，TF Encrypted 使用时与 TensorFlow 类似，利用了 Keras API 的易用性，同时通过 MPC 和 HE 对加密数据进行训练和推理。TF-Encrypted 默认使用的 MPC 协议是 pond 协议(3 方计算协议)，同时也支持 SecureNN 和ABY^3协议。TF Encrypted 专注于其低级接口，与 TensorFlow 紧密集成，并且支持第三方库，可以添加新的协议和技术。

4) PySyft

PySyft 是用于隐私保护深度学习的开源库，由 OpenMined 社区在 2019 年发布。PySyft 将联合学习、差分隐私（differential privacy，dp)、MPC（实现了 SPDZ 和 SPDZ-2 协议)结合在一个编程模型中，集成到不同的深度学习框架(如 PyTorch、Keras 和 TensorFlow）中，将隐私数据与模型训练分离。PySyft 最初基于 PyTorch，可以利用 Keras API 调用 TF Encrypted，在新的版本中 PySyft 增加了对 Tensorflow 的支持，能够方便地与 TensorFlow API 以及 TF Encrypted 等框架进行集成。PySyft 和 TF Encrypted 是两个社区驱动的项目，虽然 PySyft 的发布时间晚，但目前 PySyft 比 TF Encrypted 具有更好的生态和规模。

5) CrypTen

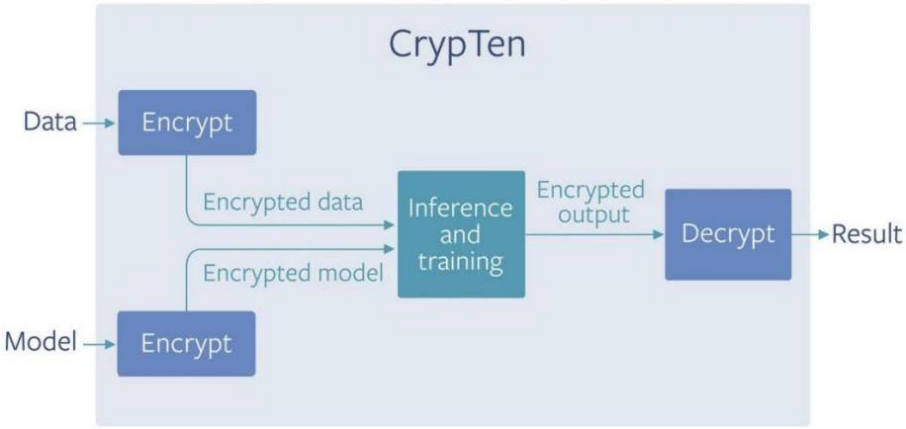

图 28　CrypTen 框架

CrypTen 是 Facebook 于 2019 年开发的基于 PyTorch 的隐私保护机器学习开源框架。CrypTen 当前实现了一种 MPC 的加密方法，满足半诚实安全模型，计划在将来的版本中增加对 HE 和安全 Enclave 的支持。CrypTen 允许用户使用类似于 PyTorch 中的自动区分和神经网络模块进行机器学习的计算，并实现了类似 PyTorch 的张量库，允许用户来构建机器学习模型。即使不了解密码学的开发人员也可以轻松地使用 CrypTen 来训练和测试机器学习模型。图 28 展示了 CrypTen 框架，其中数据和模型都使用 MPC 进行加密，使用密文数据对密文模型进行训练和推理后输出密文结果，解密后可以得到明文结果。

6)　Rosetta

Rosetta 是 Lattice 基金会于 2020 年开发的一个基于 TensorFlow 的隐私计算开源框架。Rosetta 的目的是为机器学习快速提供隐私保护技术解决方案，机器学习开发者不需要掌握关于密码学等专业知识即可进行隐私保护下的机器学习训练，而密码学研究人员也可以使用其底层 C/C++实现的高性能密码算法进行开发。Rosetta 当前版本支持 MPC 协议，默认的底层协议是三方 SecureNN，之后将陆续集成其他的 MPC 协议。Rosetta 在用户接口层复用了 TensorFlow 的对外 API，使得用户可以以最低的改造成本将隐私保护功能集成到现有的 TensorFlow 程序中。

7)　SecurePlus

Duality 公司开发了 SecurePlus 产品，该产品核心技术是同态加密，它使得数据在生命周期中总是处于安全状态，敏感数据或机器学习模型不会泄露。目前 Duality 公司已经基于 SecurePlus 平台发布了一系列 SecurePlus 产品套件，如 SecurePlus Query 和 SecurePlus Insights 等。SecurePlus 平台主要有三种应用场景：安全数据分析、机器学习模型的版权保护、数据共享的隐私保护。

a)　安全数据分析。SecurePlus 平台使得数据所有者可以对加密数据使用第三方分析工具(机器学习、数据挖掘工具)进行分析和处理。

b)　机器学习模型的版权保护。SecurePlus 平台使得模型拥有者可以在不可信第三方部署、存储和使用密文模型。通过同态加密运算，样本数据和模型可以在密文域进行推理和分类任务。

c)　数据共享的隐私保护。SecurePlus 平台确保多方数据安全共享与协作。同态加密在链接和计算过程中保护了每一方的资产，在整个过程中保护隐私。

4.2.3 其它应用系统

1) 密钥管理系统

Unbound 公司提供基于软件的安全密钥管理器，作为硬件安全模块的替代。该公司将 MPC 技术用于密钥管理中，主要采用秘密分享协议加和机制，将密钥拆分成两份分别存储在不同的机器中，在整个密钥生命周期中都无需将两个份额组合到一起。Unbound tech 发布的开源库 Blockchain-crypto-mpc 解决了区块链中的种子密钥和签名密钥的保护问题。此外，Curv 公司使用 MPC 协议消除区块链中因私钥可能引发的单点故障问题，Sepior 公司使用门限 SS 对密钥进行拆分并存储于多个服务器来防止密钥被盗以及滥用。

2) PSI 应用系统

谷歌和 Microsoft 利用 MPC 技术，在不泄露用户隐私的前提下实现了口令检查和口令监视。谷歌在 Chrome 扩展程序中的“口令检查”功能是基于 PSI 协议设计的，允许用户将其登录凭据与 40 亿个受损凭据的加密数据集进行匹配，而不会向包括 Google 在内的任何人透露详细信息。Microsoft 在 Edge 浏览器引入“口令监视器”的功能，对比存储在云中的已知泄露口令的大型数据库和用户在浏览器中保存的口令，使用同态加密对用户名和口令加密，同样使用同态加密对泄露口令数据库中的数据加密，对比同态密文来判断用户信息是否泄漏。

3) PIR 应用系统

隐私信息检索(Private Information Retrieval - PIR)技术是保护用户查询隐私的方案，当用户在数据库上检索信息时，在数据库服务器不知晓用户查询语句的相关信息的情况下完成检索。PIR 技术应用于隐私保护位置查询、金融领域的征信查询、医疗领域的治疗方案查询等实际场景。

SealPIR[132]是 2018 年提出的处于研究阶段的库，还没有应用到实际场景中，该库最新维护时间是 2022 年。SealPIR 允许客户端从服务器存储的数据库中下载元素，而无需透露下载了哪个元素。SealPIR 依赖于 Microsoft SEAL，在编译 SealPIR 之前需要安装 SEAL。XPIR[133]是 2016 年提出的隐私信息检索库，依赖于同态加密库 NFLib。

4.3 MPC 协议与工具的开源实现

随着 MPC 技术的不断发展，目前已有一些 MPC 协议或工具的开源实现方案，可以为研究人员、应用人员提供 MPC 协议的验证或 MPC 应用的快速实现环境[134]。本节从 GC、OT、SS、HE 几个技术方分别介绍现有的开源实现。现有的 GC 开源实现有 Fairplay、EMP-toolkit、Obliv-C、ObliVM，OT 的开源实现有 LibOTe、Apricot 等，SS 开源实现有 Wysteria、PICCO，HE 的开源实现有 SEAL、Helib 等此外，还有支持多种技术结合的开源实现，如，SCALE-MAMBA、ABY 等。

4.3.1 GC 开源实现

1) Fairplay

2004 年发布的 Fairplay[135]是第一个通用 MPC 系统工具，用于计算 GC 布尔电路，可以支持两方参与安全计算，如基于 Key 的数据库搜索、求中值等。Fairplay 提供了高级编程语言 Secure Function Definition Language(SFDL)及相应的编译器，编译器可将 SFDL 编译为用于描述布尔电路的、面向硬件的低级程序语言 Secure Hardware Definition Language (SHDL)。Fairplay 的工作流程如图 29 所示，其中 Alice 与 Bob 分别代表两个参与方。

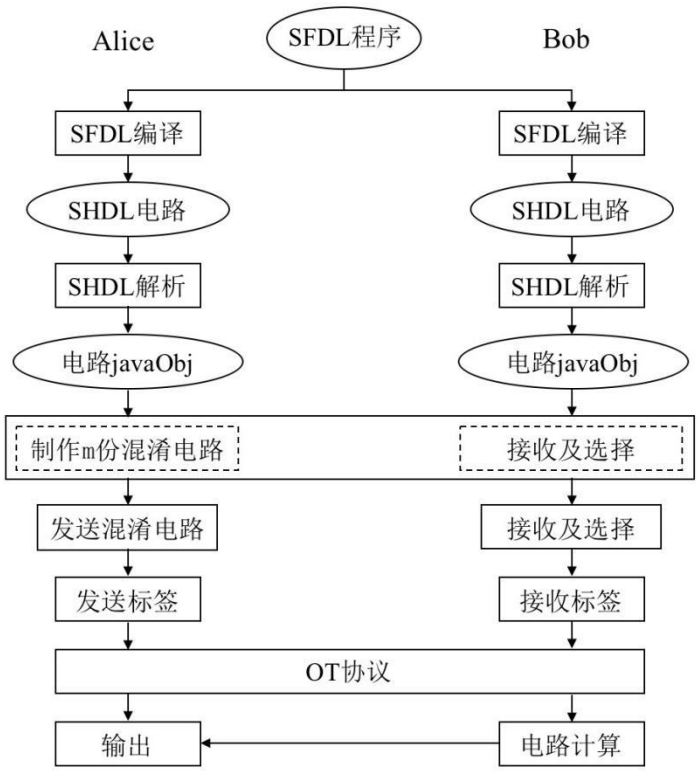

图 29　Fairplay 工作示意图

多方安全计算程序使用 SFDL 开发，经 SFDL 编译器编译后形成 SHDL 电路文件。Bob、Alice 将 SHDL 电路文件进行解析生成 java 对象，后续可执行 MPC 协议。Fairplay 支持 Cut-and-Choose 功能，可用于实现恶意安全的协议方案。FairplayMP 系统在 Fairplay 系统方案基础上，实现对于多于两个参与方的多方安全计算支持，SFDL 升级为 SFDL 2.0。Fairplay 对每一个门电路的操作都需要各方进行数据份额交互，导致执行性能很低。自 2004 年发布起，Fairplay 的版本未更新过，但在 2008 年发布了一版适合多方计算的 FairplayMP。

2)　EMP-toolkit

EMP-toolkit[136]是 2016 年发布的基于 GC 电路的 MPC 开源工具，直到目前都在一直更新源码功能。EMP-toolkit 既允许密码学研究人员使用 EMP-toolkit 底层的密码学相关的构建块来创建新的协议，也允许不了解底层加密技术的开发人员使用工具中已实现的协议开发应用程序。EMP-toolkit 的代码主要由 C++编写，其中基于密码学的基本构建块（如伪随机数发生器、散列函数等）和 MPC 协议常用的模块（如 GC、OT 等）都被实现为具有调用接口的 EMP 对象，支持开发者使用 C/C++进行调用和开发。该组件现已实现半诚实安全两方协议、恶意行为假设两方协议[25]、恶意行为假设多方协议[27]、PVC[24]以及 Ferret[44]等。

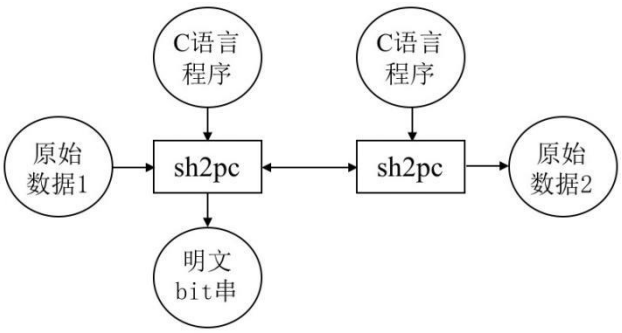

图 30　EMP-toolkit 半诚实安全模型下两方工作示意图

EMP-toolkit 提供了 sh2pc（Semi-honest Two Party Computation）库，用于处理半诚实安全模型下的两方安全交互，如图 30。两方分别运行 sh2pc，一方作为 Garbler，另一方作为 Evaluator。两方输入相同的程序，sh2pc 将之转化为特定格式的电路。

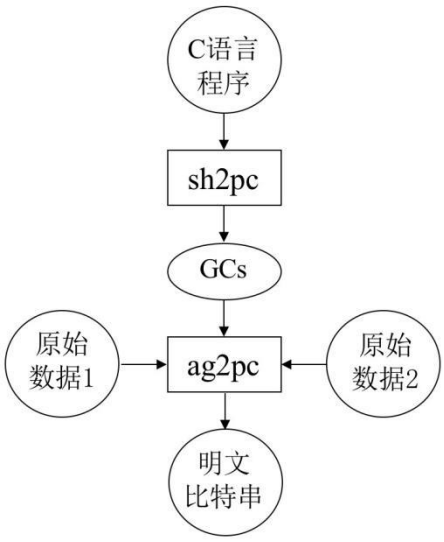

图 31　EMP-toolkit 恶意安全模型下两方工作示意图

对于恶意安全模型下的两方或多方运算，需要先通过 sh2pc 库创建电路，作为文件传递给 Auth-GC 协议的 ag2pc 库（或 agmpc 库）执行，如图 31。EMP-toolkit 的执行结果只有一方可以获得。

在半诚实安全模型下，EMP-toolkit 允许开发者使用 C 语言的数组和结构体作为隐私输入，支持任意大小的整数和浮点数，电路是在计算时产生。在恶意安全模型下，EMP-toolkit 限制输入输出为布尔数组。

3)　Obliv-C

Obliv-C[137]是于 2015 年提出的 C 语言扩展和配套编译器，支持半诚实安全模型下两方 GC 协议，其开源代码也一直保持更新状态。Obliv-C 是对 C 语言的一种扩展，增加了 Obliv 修饰词，修饰数据类型和函数。Obliv-C 的编译器将这种扩展语言转化为可执行的电路。Obliv-C 的设计目的是让开发者可以方便地利用这种编程语言来开发安全协议，同时帮助学术研究者创建自定义的库。

Obliv-C 是对 C 的扩展，但是许多应用实例都将 OblivC 代码与 C 代码分开使用，使用 C 代码读取、处理、输出数据，仅在 Obliv-C 中执行安全计算。Obliv-C

可应用于线性递归、隐私保护机器学习、加密邮件分类等。Obliv-C 维护底层协议提供的所有安全属性，保证程序执行时不会泄露任何信息，同时将所有数据无关的计算公开给开发者，在保证数据安全的前提下给予开发者足够多的控制权。

4) ObliVM

ObliVM[138]是 2015 年发布的一款开源编译器，能够编译一种类似 Java 的语言，称为 ObliVM-lang。ObliVM 将源代码转换成两方混淆电路协议，它实现了内置的高效 ORAM 方案。ObliVM 支持固定大小的整数，并包含一个支持任意大小整数的库。开发者可以在 ObliVM-lang 上实现低级的电路库，也可以在后端实现自定义的协议，并将其作为原生类型和原生函数导出到 ObliVM-lang 源语言里。ObliVM 提出的目标是提供一种易编程的语言，并高效地编译成安全计算协议，但是由于 ObliVM-lang 语言以及常规用法的文档都是有限的，I/O 也有限制，且其近年来未更新代码，导致其在实际中的应用受到了限制。

5) Frigate

Frigate[139]是于 2016 年被提出的开源电路编译器，可以将类 C 的高层语言编译为任意数量输入的自定义布尔电路。Frigate 的源码一直在更新，最近一次是在 2020 年被作者添加了新功能。Frigate 采用的电路格式能够最小化文件大小，其中包含的解释器可以在生成的电路与其它应用程序之间建立接口，方便使用者进行扩展。Frigate 支持三种数据类型：有符号整数、无符号整数、结构体，定义了比较和按位运算。Frigate 与 Obliv-C 和 ObliVM 等编译器的区别在于其可以进行严格的验证测试来检查编译器的状态，避免出错，保证正确性，但是 Frigate 的正确性检查都需要单独的后端。Frigate 可以快速生成电路，但是将电路形式连接到后端转换工具来执行端到端 MPC 计算需要繁重的操作。

6) CBMC-GC

CBMC-GC[140]是 2012 年被提出的用于安全两方计算的 ANSI C 开源编译器，能够将 C 程序转换为符合混淆电路要求的布尔电路，由该电路在两方之间执行安全计算。CBMC-GC 基于 CMBC 实现，CMBC 是一种有界模型检查器，可将任何 C 程序转换为布尔约束，然后调整该工具的输出以产生用于 MPC 计算的优化电路。CBMC-GC 包括一个运行 ABY 电路的工具，还包括一个输出网表格式电路的工具。

在 2014 年，CBMC-GC 的作者团队[141]提出了一个升级版本 CBMC-GC v0.9，添加了新的电路优化技术，能够编译生成体量更小的电路。

在 2018 年，Technische Universität Darmstadt 的安全工程组创建了 CBMC-GC v2.0 alpha，包含 CBMC-GC 和 ShallowCC[142]。ShallowCC 是编译器的扩展，可为 ANSI-C 的 MPC 编译深度优化的电路。

4.3.2 OT 开源实现

1) LibOTe

LibOTe 是一个快速且可移植的 C++库，源码近期仍在更新和维护中。LibOTe 提供了几种不同的 OT 协议实现：Naor Pinkas 的基础 OT 协议，半诚实安全模型下的 1-out-of-2 OT 实现有 IKNP03、BCG+19b[50]、BLNNOOSS15[143]、1-out-of-N OT 的实现有 KKRT16。在恶意安全模型下，1-out-of-2 OT 的实现有 KOS15、BLNNOOSS15、CO15[144]、MR19[145]，1-out-of-N OT 的实现有 OOS16，K-out-of-N OT 有 RR16。该库支持跨平台使用，并且已经在 Windows、MAC 和 Linux 上进行了测试。

2) EMP-toolkit

EMP-toolkit 在其包含的 OT 库中目前实现了两个基本 OT、IKNP03 和 Ferret 两种 OT 扩展。

3) Encryptogroup-OTExtention

Encryptogroup-OTExtension 是 2013 年发布的 C++库，该库中实现的基础 OT 方案有 NP01、CO15 和 PVW08[146]，实现的 OT 扩展方案有 ALSZ13[147]、ALSZ15、NNOB12 和 KK13[34]。

4) Apricot

Apricot 是 2016 年发布的 C++库，实现了恶意安全模型下的 OT 扩展协议 KOS15。

4.3.3 SS 开源实现

1) Wysteria

Wysteria[148]是于 2014 年提出的框架，目前已无人维护。Wysteria 基于 GMW 协议，支持任意数量的参与方。Wysteria 支持前端高级语言编程、类型检测器、运行时解释器（执行 GMW 布尔电路），并支持自然数、布尔值、数组等多种数据类型和函数计算。

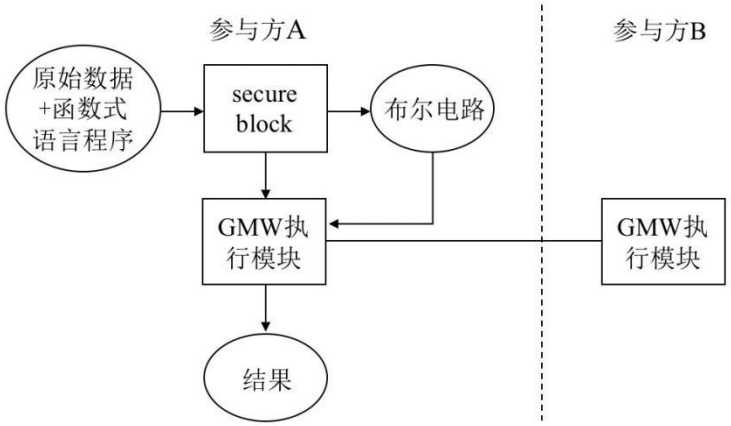

图 32 Wysteria 工作示意图

Wysteria 计算的电路可以提前生成，支持本地隐私计算和多方安全计算，工作原理如图 32。多方共同执行相同的逻辑，计算前进行数据配置，并将程序逻辑和数据输入到一个 secure block。Secure block 将程序逻辑转化为布尔电路并输出给 GMW 执行模块。各方 GMW 执行模块相互协同完成计算并共同决策一个输出结果接收方。

Wysteria 已经应用与隐私集合求交、拍卖、最近邻居求解等问题。但是 Wysteria 支持的数据类型有限，后端电路解析器也已过时，不适用于复杂的程序开发。

2) PICCO

PICCO[149]是于 2013 年提出的框架，目前仍在维护。PICCO 基于 Shamir 秘密分享，支持任意数量的参与方。PICCO 提供了一种扩展 C 语言编程接口，将用户程序转化为 C，编译后输出给 n 个服务器去运行一个多方计算。PICCO 包含三种参与方：输入提供方、计算服务器和输出接收方。PICCO 支持整数和浮点数运算，支持加减乘除等算术运算，XOR、AND、OR 等布尔运算，以及比较等逻辑运算。

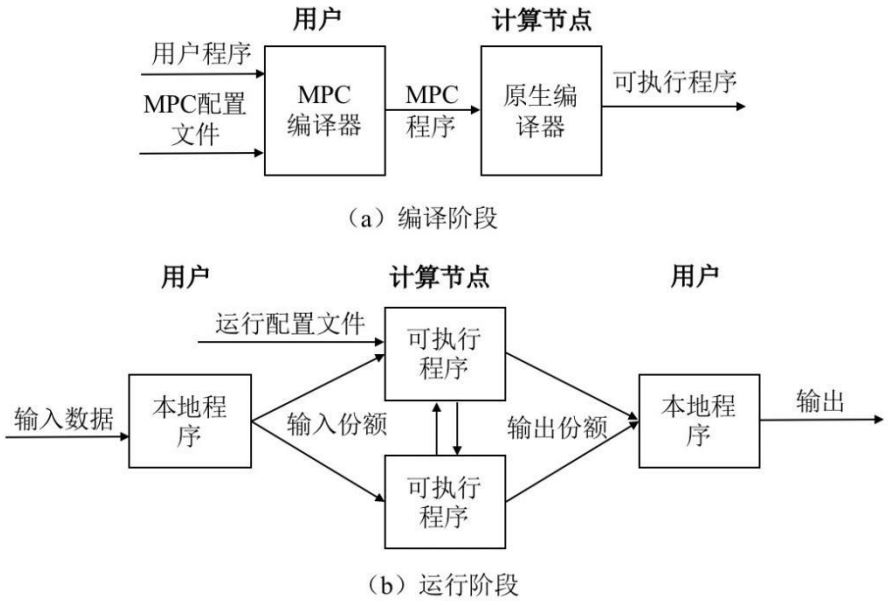

图 33　PICCO 工作示意图

PICCO 的计算过程分为两部分：编译阶段和运行阶段，如图 33。

在编译阶段，用户在 C 语言源程序中对隐私输入数据进行了标记。输入的配置文件是 MPC 的参数，编译器先将扩展的 C 语言程序编译成原生 C 程序，然后进一步编译为 MPC 可执行程序。

在运行阶段，用户将编译好的可执行程序传递给每一个计算方。启动时，用户通过本地程序将输入数据按照 SS 协议转化为份额输入给计算器，同时计算方获得运行时配置文件。图中只有一个用户提供输入、一个用户接收输出、多个计算器进行计算，实际上多个输入方必须同时进行编译、分别进行隐私数据输入。

3)　FRESCO

FRESCO 是于 2015 年提出的用 java 实现的框架。FRESCO 基于 Shamir 秘密分享，包含了 Dummy、TinyTable、SPDZ、SPDZ_{2^k}等协议套件，同时支持使用不同语言进行新协议套件或应用的开发。它将电路结构表示为代码，支持每轮对固定数量的门进行并行评估，减少大型电路中过多内存使用。FRESCO 不支持小数运算。

4)　MPyc

MPyc 是于 2018 年推出的 MPC 的 python 开发包。MPyc 提供了可以对秘密份额进行计算的运行时，参与方之间通过点对点信道进行通信。MPyc 实现了有限域上的 Shamir 秘密共享方案，可以容忍少数不诚实参与方，支持加减乘除、固定点算术运算、比较大小和比特级运算。

4.3.4 HE 开源实现

1) SEAL

SEAL 是由 Microsoft 开发的 C++同态加密库。SEAL 实现了 BFV 和 CKKS 两种同态加密方案，支持 Windows、Linux、MACOS、FreeBSD、Android 等操作系统平台，同时支持.NET 开发。SEAL 支持扩展，提供了 C++、C#/F#、Python、JavaScript、TypeScript 等开发语言的编程接口。SEAL 的可选依赖项有 GSL、ZLIB 和 Google Test 等第三方库。在噪声管理方面，与 HElib 支持自动噪声管理不同，

在 SEAL 中每个密文拥有一个特定的噪声预算量，需要在程序编写过程中通过重线性化操作自行控制乘法运算产生的噪声。

2) HElib

HElib 是 IBM 开发的 C++同态加密库，最初版本由 2013 年发布，2018 年以来，HElib 一直在可靠性、鲁棒性、性能以及可用性方面进行广泛的重构。HElib 底层依赖于 NTL 数论运算库和 GMP 多精度运算库实现，支持在 Windows、MACOS、Linux 等操作系统平台上进行安装部署。HElib 实现了 BGV 方案和 CKKS 方案，同时在上述原始方案中引入了许多优化以加速同态运算，包括 Smart-Vercauteren 密文打包技术和 Gentry-Halevi-Smart 优化，提升了算法的整体运行效率。HElib 提供了一种“HE 汇编语言”来支持一些基本操作指令，此外还提供了自动噪声管理、改进的 Bootstrapping 方法、多线程等功能。

3) PALISADE

PALISADE 是由新泽西理工学院、Duality 等公司开发的 C++格密码库。该库首次发布于 2017 年，稳定版本 v1.10.6 于 2020 年 12 月发布。PALISADE 实现了 BGV，BFV，CKKS、FHEW 和 TFHE 全同态加密方案，此外还实现了多方全同态加密方案：BGV、BFV、CKKS 的门限 FHE 方案以及代理重加密方案。PALISADE 支持 Windows、MACOS、Linux 平台，提供了 C++、Python、FreeBSD 的可编程接口。

4) HEAAN

HEAAN 是由 CKKS 方案作者开发的一个 C++同态加密库，底层依赖于 NTL 数论运算库，它实现了 CKKS 方案。HEAAN 于 2016 年发布的第一个版本 v1.0 实现了原始的 HEAAN 算法，v1.1 版本实现了具有自举算法的版本，v2.1 是 HEAAN 的快速实现。

5) FHEW

FHEW 是主要用 C/C++编写的全同态加密库。FHEW 基于论文[96]实现并使用到 FFTW 库，支持对二进制字符串的运算，可以进行非多项式计算。

6) TFHE

TFHE 是一个 C++同态加密库，支持在常规 C 语言代码中使用，TFHE 首次发布日期是 2017 年，最新版 v1.1 于 2020 年发布，目前支持在 Linux、MACOS 平台部署。该库基于论文[67]实现了快速的逐门自举方案，实现了 GSW 方案的环上的变体，并且进行了优化。TFHE 实现了快速的二进制门同态运算，可以支持非多项式计算。与其他库不同，TFHE 对门的数量或其组成没有限制。

7) FV-NFLlib

FV-NFLlib 是基于 NFLlib 格密码库实现的 C++同态加密库，最新维护时间是 2016 年。FV-NFLlib 实现了 FV 方案，需要 GMP、Mpfr、NFLlib 库的支持，该库只包含一个头文件，可以在编译时使用。

8) Lattigo

Lattigo 是用 Go 语言实现了基于 RLWE 的同态加密方案以及多方同态方案，Lattigo v2.2.1 发布于 2020 年。包括 BFV 及 CKKS 方案以及它们的多方同态版本。Lattigo 旨在在分布式系统和微服务架构中支持同态加密，可实现跨平台构建，具有与最新 C++同态库可比的性能。目前该库仍处于实验阶段，仅用于研究。

9) cuHE

cuHE （CUDA 同态加密库）是 HE 的 GPU 加速库，用于多项式环上定义的同态加密方案，最新维护时间是 2017 年。cuHE 利用 GPU 提供的大规模并行性和高带宽的特性，对内存最小化、内存及线程调度等进行优化，使用快速数论变换

（NTT）、中国剩余定理（CRT）、Barrett reduction 来处理多项式函数。cuHE 库仅出于研究目的而创建，可以快速开发高性能应用程序，在[155]等论文中有对该库算法和优化的说明。

10) CHET

CHET[156]是 Microsoft 于 2018 年提出的用于简化 FHE 应用的编译器，最新维护时间为 2020 年。CHET 用于优化 HE 在神经网络推理过程中的应用，可以自动对张量进行同态加密（包括选择合适的加密参数来满足安全性和准确性），并且能够切换不同的同态加密方案，CHET 可以将张量编译为链接到 SEAL 或 HEAAN 的可执行文件。

11) nGraph-HE

nGraph-HE[157]是 Intel 于 2019 年提出的对深度学习中 HE 算法进行优化的图编译器，最新维护时间是 2020 年。nGraph-HE 可以在编译时使用 HE-SIMD 打包、OpenMP 并行、以及对 avg pooling 和 Batch-Nomalization 中的同态计算进行优化，在运行时对特殊值绕过而减小计算开销。nGraph-HE 支持 SEAL 库中的 CKKS 方案，为了计算激活函数与 ABY 库集成，还与 Tensorflow 运行引擎集成来让用户在使用 Tensorflow 训练神经网络时调用 nGraph-HE。

4.3.5 支持多种技术方案的开源实现

1) ABY

ABY[150]是于 2015 年提出的一个混合协议两方计算框架，以 C++库的形式实现。它通过提供一种混合协议的机制，为开发人员提供对计算效率的细粒度控制。ABY 目前支持三种协议，并可以互相转换：基于 Beaver 三元组的 Arithmetic 运算；基于 GMW 协议的 Boolean 运算和基于优化的 Yao's GC 协议。

在 ABY 中，安全数据仅限于无符号 C 整数类型，且其不支持任意长度的整数或布尔类型。ABY 将安全数据存储在 C 结构中，并支持 C ++数组和 SIMD 结构以实现有效的并行操作。此外，ABY 支持一些浮点运算，并且正在积极开发此功能。

ABY 提供了功能强大的低级密码接口，可让开发人员有效地控制性能。 ABY 面向熟悉 MPC 协议和计算电路模型的用户，适合有密码背景的开发人员。

2) $\mathbf{ABY^3}$

$\mathbf{ABY^3}$[112]于 2018 年被提出，是一个隐私保护机器学习的通用框架，目前已有开源的 C++实现，并在近期进行了更新。$\mathbf{ABY^3}$使用的协议是一个三服务器模型，能够在 3PC 的基础上实现隐私保护和模型训练。$\mathbf{ABY^3}$是建立在 libOTe 库和线性代数库 Eigen 提供的原语基础上的，并且所有的算术部分都是以2^{64}为模进行的，可以高效地在算数共享、二进制共享和 Yao 3PC 间进行切换。

3) SCALE-MAMBA

SCALE-MAMBA 实现了 MPC 混合协议，取代了 SPDZ 框架，其支持 GC（BMR）和 SS(SPDZ)协议。SCALE-MAMBA 包括运行时系统 SCALE 和类 Python 高级语言 MAMBA。

SCALE-MAMBA 处理过程有三部分：离线过程、在线过程、编译过程。与 SPDZ 不同的是，SCALE 将离线处理和在线处理集成在一起。编译过程将 MAMBA 语言程序转化为可被 SCALE 执行的字节码。SCALE-MAMBA 主要基于研究成果[57][151]等。SCALE-MAMBA 支持多个参与方，允许不诚实人数占大多数，采用基于 MAC 的方式发现恶意参与方，并在发现有恶意行为时中止运算。

为提升执行效率，SCALE 主程序在起始时生成若干个需要在线执行的线程，线程的数量由编译器决定。每个在线执行的任务绑定 4 个离线的处理线程：

Beaver 三元组生成器、比特份额生成器、square 对生成器以及输入输出预处理。在线处理过程同时从编译器获得字节码、以及输入数据进行计算，最后输出结果。SCALE-MAMBA 依托 Zaphod[152]实现 BMR 计算和 SPDZ 计算之间的转换

SCALE-MAMBA 有大量文档，涵盖了以前的 SPDZ 系统，允许开发人员定义自己的 I/O 类。虽然 SCALE-MAMBA 支持任意数量的参与方并且具有强大的安全保证，但是同时也需要很大的计算资源。

4) MP-SPDZ

MP-SPDZ 是 SPDZ-2 的一个分支，于 2020 年被提出，实现为开源框架，至目前一直有代码更新和维护。MP-SPDZ 用于对各种 MPC 协议（如 SPDZ、SPDZ2k、MASCOT、Overdrive、BMR GC、Yao's GC 以及基于三方复制的秘密共享和 Shamir 秘密共享）进行测试，MP-SPDZ 将 SPDZ-2 扩展到多个 MPC 协议变体，所有协议都可以与基于 Python 的同一个高级编程接口一起使用。MP-SPDZ 大大简化了不同协议和安全模型的比较成本。

MP-SPDZ[153]支持的协议涵盖了所有常用的安全模型（诚实/不诚实多数和半诚实/恶意破坏）以及二进制和算术电路的计算（以素数和 2 的幂为模）。所采用的基础原语包括秘密共享、不经意传输、同态加密和乱码电路。MP-SPDZ 适合于为有或没有安全计算背景的研究人员对各种安全模型中的计算成本进行基准测试。

5) HyCC

HyCC[154]是于 2018 年提出的针对 ANSI C 的开源编译器，是 CBMC-GC 和 ABY 框架的后续工作。HyCC 的优势在于其能够将用标准 ANSI C 代码编写的应用程序自动编译成混合 MPC 协议，而不是只能编译为一种或两种 MPC 协议，它能高效、安全地将多个 MPC 协议与优化编译、调度和分区结合在一起，这使得没有 MPC 专业知识的用户也能更轻松地进行开发。HyCC 将 ANSI C 程序编译为布尔和算术电路，再将电路优化，最后对于给定的部署场景（如最小延迟、通信成本等）选择合适的 MPC 协议。HyCC 混合编译方面的工作是建立在 CBMC-GC 上的，而电路评估部分是基于 ABY 进行的。

5. 知识产权与标准发展

国内外有较多投入 MPC 研究的公司，国外如 visa、IBM、unbound、Microsoft 等，国内如阿里巴巴、蚂蚁金服、腾讯、百度、京东等，新兴技术公司如华控清交、矩阵元等。这些公司进行了技术、产品研发及知识产权（专利）的申请，已布局的 MPC 系统产品的方面如：基础协议（通用计算/专用计算）、MPC 安全验证技术和组合技术、MPC 相关硬件、MPC 系统安全、调度管理、数据处理、机器学习算法实现、调试工具、产品系统和行业解决方案等方面。

5.1 专利

基于智慧芽(https://analytics.zhihuiya.com/)的专利检索全球数据库，以“多方安全计算、安全多方计算、multi-party computation、multi-party calculation、multiparty computation、multi-party computation、secure multiparty computation、secure multi-party computation”等作为关键词进行检索，得到近年来 MPC 相关专利的发展趋势（申请数量）：

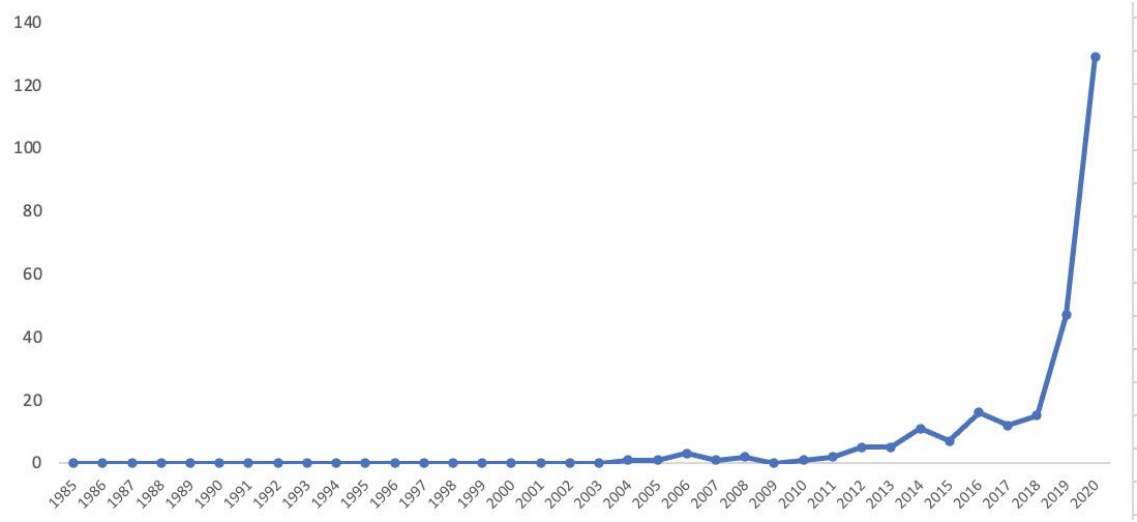

图 34　国外专利发展趋势

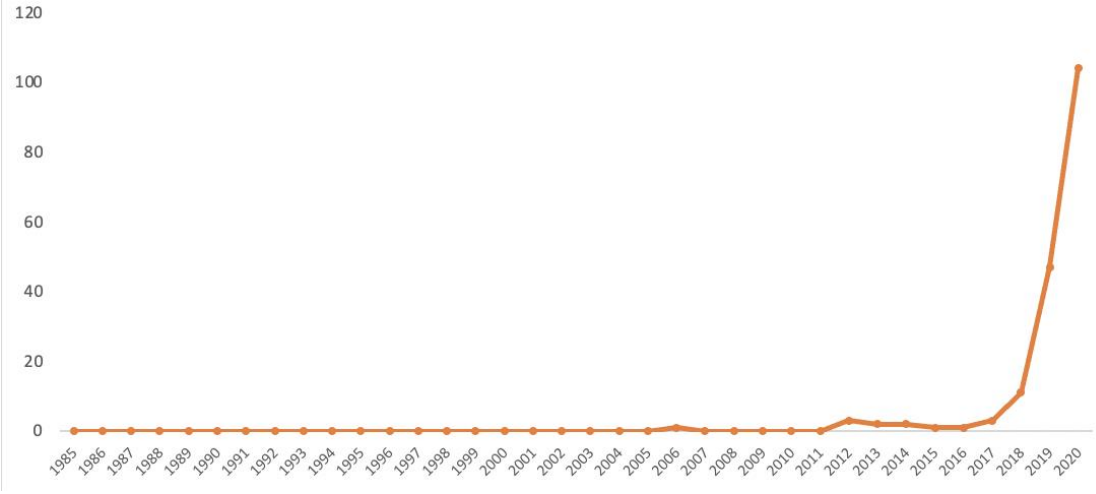

图 35　国内专利发展趋势

从发展趋势上看，2000 年之后才逐步出现 MPC 相关专利，并且近几年才有显著发展。特别是最近 3 年的专利申请数量呈现井喷式发展。

国内专利在总量上基本与国外相当，发展趋势一致，说明 MPC 技术的产业化发展得到国内外的普遍重视。

5.2 国外 MPC 相关标准

5.2.1 秘密分享技术标准

国际标准组织 ISO/IEC 发布了秘密分享技术系列标准 ISO/IEC 19592。ISO/IEC 19592-1 为标准第一部分，发布于 2016 年，标准对秘密分享技术给出了通用的基本术语与定义指定了密码秘密分享方案及其属性，提出秘密分享方案的一般模型，包括参与方、参数、消息空间、份额空间、份额数量、秘密分享过程、秘密恢复过程，对于秘密分享方案涉及的属性给出说明，包括消息机密性、消息可恢复性、同态性、可验证性、复杂性等。ISO/IEC 19592-2 为标准第二部分，发布于 2017 年，介绍了 5 种秘密分享方案，包括 Shamir 秘密分享、ramp Shamir

秘密分享、加法秘密分享、冗余的加法秘密分享等。对每种算法分别描述了参数设置、秘密分享算法、秘密恢复算法、属性等。

美国国家标准与技术研究所（NIST）已着手开展秘密分享技术的标准化研究工作，并发布分别与 2019 年 3 月、2020 年 6 月发布了秘密分享技术系列技术报告 NISTIR 8214《Threshold Schemes for Cryptographic Primitives》[158]、NISTIR 8214A《NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives》[159]。NISTIR 8214 对门限方案属性进行分析，包括机密性、可用性、完整性以及它们之间的关系；提出门限方案的系统模型，包括参与方实体、接口、身份信任和分布式共识；对门限方案特征进行分析，包括门限值、通信模型；对目标计算平台进行总结，包括硬件、软件、单设备、多设备和辅助组件；分析如何设置和维护系统，包括节点的部署，有无 Dealer 的初始化设置等。NISTIR 8214A 分析了门限方案中可以进行标准化的内容，包括两种场景（单设备、多设备）、密码原语、输入输出接口的模式、客户端的互操作性和可审计性;讨论了在密码系统中实现门限方案的动机,例如防止单点故障、增强可用性等；列出了需要标准化的内容，包括签名、加密、随机数生成等；分析了标准化内容的属性，包括对算法和模块的验证、配置细节和安全功能、模块化等。

5.2.2 同态加密技术标准

国际标准化组织（ISO）于 2019 年发布了同态加密标准 ISO/IEC 18033-6:2019 《IT Security techniques — Encryption algorithms — Part 6: Homomorphic encryption》[160]。该标准定义了同态加密的基本模型，并对两种较为成熟的半同态加密机制进行了描述，包括 ElGamal 乘法同态加密和 Paillier 加法同态加密。针对以上两种同态加密机制，分别规定了参与实体的参数和密钥生成、数据加密、密文数据解密、密文数据同态运算等步骤的具体过程。

同态加密标准化开放联盟（HomomorphicEncryption.org）是由来自产业、学术、政府等机构成员构成，致力于同态密码技术的标准化工作，其成员包括 Microsoft、Intel、IBM、Alibaba、NIST、ITU、MIT 等。联盟于 2018 年发布了《Homomorphic Encryption Standard》文件[161]。该文件包含两个部分，第一部分对经典加密方案进行规范：给出通用的术语和定义；定义了安全属性；介绍了 BGV、BFV、GSW 方案；提及可替代方案 YASHE13[162]、HPS98[163]/LTV12 和 CKKS；最后讨论了分布式 HE、侧信道攻击、密钥评估等其它特征。第二部分对实现安全性的参数给出建议：描述 LWE 和 RLWE 安全假设；介绍已有的格攻击、LWE 上的 Arora-Ge 攻击、RLWE 上的代数攻击；最后对实现不同级别的安全性所需的参数提出建议。除上述标准文件外,联盟还发布了同态加密安全白皮书[164]、API 白皮书和相关应用白皮书[165]。同态加密 API 白皮书[166]介绍了基于 RLWE 的同态加密方案的 API 设计要求；相关应用白皮书介绍了同态加密的潜在应用，包括：在基因组学中的应用、国家安全/关键基础设施中的应用、教育应用、医疗保健应用、保护控制系统应用等。

5.2.3 MPC 标准

自 2019 年起，国际电信联盟（ITU）、电气和电子工程师协会（IEEE）、ISO 等国际标准组织均已开展 MPC 技术标准研制工作，目前均处于标准制订阶段。我国的技术企业及研究机构在 MPC 的国际标准制订中起到了重要作用。

国际电信联盟通信标准局安全研究组（ITU-T SG17）于 2019 年 8 月通过了《Technical Guidelines for Secure Multi-Party Computation》国际标准立项，该标准由阿里巴巴牵头制订。2021 年 10 月，该标准批准为 ITU-T X.1770《Technical Guidelines for Secure Multi-Party Computation》。

电气和电子工程师协会（IEEE）于 2019 年 10 月通过《Recommended Practice for Secure Multi-party Computation》国际标准立项。该标准由阿里巴巴牵头制订，当前标准项目号为 IEEE P2842。该标准内容主要包括 MPC 技术框架、安全级别、基于 MPC 的实现用例等。该标准于 2021 年 11 月正式发布。

ISO 于 2020 年 6 月启动 MPC 标准制订工作，标准号为 ISO/IEC 4922。ISO/IEC 4922 目前包含 2 部分，ISO/IEC 4922-1《Information security -- Secure multiparty computation Part1: General》，具体规定 MPC 基本概念、安全模型、参与方、输入输出、参数等；ISO/IEC 4922-2.2《Information security - Secure multiparty computation Part 2: Mechanisms based on secret sharing》，对基于秘密共享方案实现的 MPC 机制进行标准规定。目前，ISO/IEC 4922-1 于 2023 年 7 月正式发布；ISO/IEC 4922-2 于 2024 年 3 月正式发布，ISO/IEC 4922-2 于 2024 年 3 月正式发布。

5.3 国内 MPC 相关标准

2016 年，工信部发布的《大数据产业发展规划（2016-2020 年）》中关于提升大数据安全保障能力的重点任务，多方安全计算被作为重点研究内容提出，推动了我国在多方安全计算方面的技术研究、应用推广及标准制订工作。2018 年 5 月，中国信息通信研究院（简称“信通院”）发布了《数据流通关键技术白皮书》，以多方安全计算为例研讨了数据流通中的安全技术框架，并对多方安全计算的基本原理、安全特性等进行了描述。工信部信通院、TC601 大数据技术标准推进委员会于 2019 年 12 月发布《多方安全计算技术与应用研究报告》，在对 MPC 技术进行介绍的基础上，重点描述了几类典型应用场景，并分析了目前 MPC 技术的测评现状，最后给出了 MPC 技术应用发展的挑战和未来趋势。2020 年 11 月，信通院、阿里巴巴、数牍科技联合发布了《隐私保护计算技术研究报告》，报告中描述了 5 种隐私保护计算关键技术，多方安全计算作为其中之一被提出。该报告对于多方安全计算的安全模型、安全属性、关键技术等进行了介绍，并着重介绍了多方安全计算用于解决隐私保护问题的解决方案。

目前，我国在密码算法、大数据及金融领域已开展 MPC 相关标准的研究工作，主要以应用标准为主。

深圳市商用密码行业协会于 2019 年 11 月发布了团体标准 T/SCCIA 001-2019《基于整数同态加密的密文查询算法》，主要包括加密算法和密文查询算法两部分。该标准中提出基于整数同态加密的密文查询算法：数据加密采取混合加密的方式，即文件主体采用对称加密算法加密，文件关键字采用同态加密算法加密；密文查询只需对文件关键字的密文进行一次同态减法计算即可实现。

中国通信标准化协会大数据技术标准推进委员会（CCSA TC601）于 2019 年 6 月发布了《基于多方安全计算的数据流通产品技术要求和测试方法》标准。该

标准规定了基于多方安全计算的数据流通产品必要的技术要求和相应的测试方法，适用于基于多方安全计算的数据流通产品的研发、测试、评估和验收等。该标准的测试要求已增加至“大数据产品能力测评”，截止 2020 年 12 月已有国内多家企业的 MPC 产品已经基于该标准完成了检测认证。《基于多方安全计算的数据流通产品技术要求和测试方法》标准已立项通信行业标准制订。

中国人民银行于 2020 年 11 月发布了金融行业标准 JR/T 0196-2020《多方安全计算金融应用技术规范》。该标准主体内容包括基本功能要求、安全性要求、性能要求，旨在保障金融业各参与方的数据隐私安全，保障计算和分析的准确性，引导、规范、促进多方安全计算技术在金融行业广泛的、安全的应用。目前基于该标准的金融检测认证工作正在进行中。

目前，中国通信标准化协会仍有两项通信行业标准处于制订过程，分别是由阿里巴巴牵头的《基于安全多方计算的隐私保护技术指南》（送审）、由信通院牵头的《大数据 基于安全多方计算的机器学习技术要求与测试方法》（征求意见）。《基于安全多方计算的隐私保护技术指南》提出了 MPC 技术架构、通用流程及安全分类，给出了 MPC 典型应用场景。《大数据 基于安全多方计算的机器学习技术要求与测试方法》规定了基于联邦学习的数据流通产品必要的技术和相应的测试方法。

参考文献

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

[2]. Yao A C C. How to generate and exchange secrets[C]//27th Annual Symposium on Foundations of Computer Science (sfcs 1986). IEEE, 1986: 162-167.

[3]. 李强, 颜浩, 陈克非. 安全多方计算协议的研究与应用[J]. 计算机科学, 2003, 30(008):52-55.

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

[5]. Beaver D. Efficient multiparty protocols using circuit randomization[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1991: 420-432.

[6]. Paillier P. Public-key cryptosystems based on composite degree residuosity classes[C]//International conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 1999: 223-238.

[7]. Pinkas B, Schneider T, Smart N P, et al. Secure two-party computation is practical[C]//International conference on the theory and application of cryptology and information security. Springer, Berlin, Heidelberg, 2009: 250-267.

[8]. Kolesnikov V, Mohassel P, Rosulek M. FleXOR: Flexible garbling for XOR gates that beats free-XOR[C]//Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2014: 440-457.

[9]. Beaver D, Micali S, Rogaway P. The round complexity of secure protocols[C]//Proceedings of the twenty-second annual ACM symposium on Theory of computing. 1990: 503-513.

[10]. Naor M, Pinkas B, Sumner R. Privacy preserving auctions and mechanism design[C]//Proceedings of the 1st ACM Conference on Electronic Commerce. 1999: 129-139.

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

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

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

[14].Kolesnikov V. Gate evaluation secret sharing and secure one-round two-party computation[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2005: 136-155.

[15].Applebaum B, Ishai Y, Kushilevitz E. How to garble arithmetic circuits[J]. SIAM Journal on Computing, 2014, 43(2): 905-929.

[16].Pinkas B. Fair secure two-party computation[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2003: 87-105.

[17].Nielsen J B, Orlandi C. LEGO for two-party secure computation[C]//Theory of Cryptography Conference. Springer, Berlin, Heidelberg, 2009: 368-386.

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

[19].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.

[20].Choi S G, Katz J, Malozemoff A J, et al. Efficient three-party computation from Cut-and-Choose[C]//Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2014: 513-530.

[21].Mohassel P, Rosulek M, Zhang Y. Fast and secure three-party computation: The garbled circuit approach[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015: 591-602.

[22].Asharov G , Orlandi C . Calling Out Cheaters: Covert Security with Public Verifiability[C]// International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2012.

[23].Kolesnikov V , Malozemoff A J . Public Verifiability in the Covert Model (Almost) for Free[C]// International Conference on the Theory and Application of Cryptology and Information Security. Springer Berlin Heidelberg, 2015.

[24].Hong C, Katz J, Kolesnikov V, et al. Covert security with public verifiability: faster, leaner, and simpler[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2019: 97-121.

[25].Wang X, Ranellucci S, Katz J. Authenticated garbling and efficient maliciously secure two-party computation[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 21-37.

[26].Nielsen J B , Nordholt P S , Orlandi C , et al. A New Approach to Practical Active-Secure Two-Party Computation[M]. Springer-Verlag New York, Inc. 2012.

[27].Wang X, Ranellucci S, Katz J. Global-scale secure multiparty computation[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 39-56.

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

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

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

[31].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.

[32].Naor M, Pinkas B. Efficient oblivious transfer protocols[C]//SODA. 2001, 1: 448-457.

[33].Ishai Y, Kilian J, Nissim K, et al. Extending oblivious transfers efficiently[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2003: 145-161.

[34].Kolesnikov V, Kumaresan R. Improved OT extension for transferring short secrets[C]//Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2013: 54-70.

[35]. Kolesnikov V, Kumaresan R, Rosulek M, et al. Efficient batched oblivious PRF with applications to private set intersection[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 818-829.

[36]. Beaver D. Correlated pseudorandomness and the complexity of private computations[C]//Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996: 479-488.

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

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

[39]. Rindal P, Rosulek M. Improved private set intersection against malicious adversaries[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2017: 235-259.

[40]. Micali S, Goldreich O, Wigderson A. How to play any mental game[C]//Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC. ACM, 1987: 218-229.

[41]. Kilian J. Founding crytpography on oblivious transfer[C]//Proceedings of the twentieth annual ACM symposium on Theory of computing. 1988: 20-31.

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

[43]. Ishai Y, Prabhakaran M, Sahai A. Founding cryptography on oblivious transfer–efficiently[C]//Annual international cryptology conference. Springer, Berlin, Heidelberg, 2008: 572-591.

[44]. Yang K, Weng C, Lan X, et al. Ferret: Fast Extension for coRRElated oT with small communication[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 1607-1626.

[45]. Ishai Y, Kushilevitz E, Meldgaard S, et al. On the power of correlated randomness in secure computation[C]//Theory of Cryptography Conference. Springer, Berlin, Heidelberg, 2013: 600-620.

[46]. Damgård I, Nielsen J B, Nielsen M, et al. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited[C]//Annual International Cryptology Conference. Springer, Cham, 2017: 167-187.

[47]. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In ACM Conf. on Computer and Communications Security (CCS) 2018, pages 896–912. ACM Press, 2018.

[48]. Phillipp Schoppmann, Adri`a Gasc´on, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In ACM Conf. on Computer and Communications Security (CCS) 2019, pages 1055–1072. ACM Press, 2019.

[49]. Boyle E, Couteau G, Gilboa N, et al. Efficient pseudorandom correlation generators: Silent OT extension and more[C]//Annual International Cryptology Conference. Springer, Cham, 2019: 489-518.

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

[51]. Abascal J, Faghihi Sereshgi M H, Hazay C, et al. Is the Classical GMW Paradigm Practical? The Case of Non-Interactive Actively Secure 2PC[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 1591-1605.

[52]. Goldwasser S, Ben-Or M, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computing[C]//Proc. of the 20th STOC. 1988: 1-10.

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

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

[55].Wagh S, Gupta D, Chandran N. SecureNN: Efficient and Private Neural Network Training[J]. IACR Cryptol. ePrint Arch., 2018, 2018: 442.

[56].Li Y, Xu W. PrivPy: General and scalable privacy-preserving data mining[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 1299-1307.

[57].Damgård, Ivan, Pastro V , Smart N , et al. Multiparty Computation from Somewhat Homomorphic Encryption[C]// Cryptology Conference on Advances in Cryptology-crypto. Springer-Verlag New York, Inc. 2012.

[58].Damgård I, Keller M, Larraia E, et al. Practical covertly secure MPC for dishonest majority–or: breaking the SPDZ limits[C]//European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, 2013: 1-18.

[59].Keller M, Orsini E, Scholl P. MASCOT: faster malicious arithmetic secure computation with oblivious transfer[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 830-842.

[60].Keller M, Pastro V, Rotaru D. Overdrive: Making SPDZ great again[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2018: 158-189.

[61].Cramer R, Damgård I, Escudero D, et al.$SPDZ_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority[C]//Annual International Cryptology Conference. Springer, Cham, 2018: 769-798.

[62].Baum C, Orsini E, Scholl P, et al. Efficient Constant-Round MPC with Identifiable Abort and Public Verifiability[C]//Annual International Cryptology Conference. Springer, Cham, 2020: 562-592.

[63].Goyal V, Song Y, Zhu C. Guaranteed output delivery comes free in honest majority MPC[C]//Annual International Cryptology Conference. Springer, Cham, 2020: 618-646.

[64].Aloufi A, Hu P, Song Y, et al. Computing blindfolded on data homomorphically encrypted under multiple keys: An extended survey[J]. arXiv preprint arXiv:2007.09270, 2020.

[65].Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information[C]//STOC, 1982：365–377, 1982.

[66].Armknecht F, Sadeghi A R. A New Approach for Algebraically Homomorphic Encryption[J]. IACR Cryptol. ePrint Arch., 2008, 2008: 422.

[67].Chillotti I, Gama N, Georgieva M, et al. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds[C]//international conference on the theory and application of cryptology and information security. Springer, Berlin, Heidelberg, 2016: 3-33.

[68].李增鹏, 马春光, 周红生. 全同态加密研究[J]. 收藏, 2017, 6.

[69].Boneh D, Goh E J, Nissim K. Evaluating 2-DNF formulas on ciphertexts[C]//Theory of cryptography conference. Springer, Berlin, Heidelberg, 2005: 325-341.

[70].Gentry C. Fully homomorphic encryption using ideal lattices[C]//Proceedings of the forty-first annual ACM symposium on Theory of computing. 2009: 169-178.

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

[72].Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP[C]//Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2012: 868-886.

[73].Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption[J]. IACR Cryptol. ePrint Arch., 2012, 2012: 144.

[74].Cheon J H, Kim A, Kim M, et al. Homomorphic encryption for arithmetic of approximate numbers[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, Cham, 2017: 409-437.

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

[76].Desmedt Y, Frankel Y. Threshold cryptosystems[C]//Conference on the Theory and Application of Cryptology. Springer, New York, NY, 1989: 307-315.

[77].Pedersen T P. A threshold cryptosystem without a trusted party[C]//Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1991: 522-526.

[78].Boneh D, Gennaro R, Goldfeder S, et al. Threshold cryptosystems from threshold fully homomorphic encryption[C]//Annual International Cryptology Conference. Springer, Cham, 2018: 565-596.

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

[80].Boneh D, Gennaro R, Goldfeder S, et al. Threshold cryptosystems from threshold fully homomorphic encryption[C]//Annual International Cryptology Conference. Springer, Cham, 2018: 565-596.

[81].Badrinarayanan S , Jain A , Manohar N , et al. Secure MPC: Laziness Leads to GOD[M]// Advances in Cryptology – ASIACRYPT 2020. 2020.

[82].Chen L, Zhang Z, Wang X. Batched multi-hop multi-key FHE from ring-LWE with compact ciphertext extension[C]//Theory of Cryptography Conference. Springer, Cham, 2017: 597-627.

[83].Chen H, Chillotti I, Song Y. Multi-key homomorphic encryption from TFHE[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, Cham, 2019: 446-472.

[84].Chen H, Dai W, Kim M, et al. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 395-412.

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

[86].Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings[J]. Journal of the ACM (JACM), 2013, 60(6): 1-35.

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

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

[89].Brakerski Z, Perlman R. Lattice-based fully dynamic multi-key FHE with short ciphertexts[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2016: 190-213.

[90].Dodis Y, Halevi S, Rothblum R D, et al. Spooky encryption and its applications[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2016: 93-122.

[91].Yasuda S, Koseki Y, Hiromasa R, et al. Multi-key Homomorphic Proxy Re-Encryption[C]//International Conference on Information Security. Springer, Cham, 2018: 328-346.

[92].Li N, Zhou T, Yang X, et al. Efficient multi-key fhe with short extended ciphertexts and directed decryption protocol[J]. IEEE Access, 2019, 7: 56724-56732.

[93].Chen H, Dai W, Kim M, et al. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 395-412.

[94].Aloufi A, Hu P. Collaborative homomorphic computation on data encrypted under multiple keys[J]. arXiv preprint arXiv:1911.04101, 2019.

[95].Chen H, Chillotti I, Song Y. Multi-key homomorphic encryption from TFHE[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, Cham, 2019: 446-472.

[96].Ducas L, Micciancio D. FHEW: bootstrapping homomorphic encryption in less than a second[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2015: 617-640.

[97].秦静，张振峰，冯登国，等. 无信息泄漏的比较协议[J]. 软件学报, 2004, 3.

[98].刘文，王永滨. 安全多方信息比较相等协议及其应用[J]. 电子学报，2012，40(5): 871-876.

[99].李顺东，司天歌，戴一奇. 集合包含与几何包含的多方保密计算[J]. 计算机研究与发展, 2005, 42(10): 1647.

[100]. 张明武, 张依梦, 谌刚. 隐私保护的两方几何圆位置关系判定[J]. 密码学报, 2020, 8(1): 40-54.

[101]. Gilad-Bachrach R, Dowlin N, Laine K, et al. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy[C]//International Conference on MAChine Learning. PMLR, 2016: 201-210.

[102]. Mohassel P, Zhang Y. Secureml: A system for scalable privacy-preserving MAChine learning[C]//2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017: 19-38.

[103]. Liu J, Juuti M, Lu Y, et al. Oblivious neural network predictions via minionn transformations[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 619-631.

[104]. Chandran N, Gupta D, Rastogi A, et al. EzPC: programmable, efficient, and scalable secure two-party computation for MAChine learning[J]. ePrint Report, 2017, 1109.

[105]. Rouhani B D, Riazi M S, Koushanfar F. Deepsecure: Scalable provably-secure deep learning[C]//Proceedings of the 55th Annual Design Automation Conference. 2018: 1-6.

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

[107]. Juvekar C, Vaikuntanathan V, Chandrakasan A. {GAZELLE}: A low latency framework for secure neural network inference[C]//27th {USENIX} Security Symposium ({USENIX} Security 18). 2018: 1651-1669.

[108]. Riazi M S, Samragh M, Chen H, et al. {XONN}: Xnor-based oblivious deep neural network inference[C]//28th {USENIX} Security Symposium ({USENIX} Security 19). 2019: 1501-1518.

[109]. Agrawal N, Shahin Shamsabadi A, Kusner M J, et al. QUOTIENT: two-party secure neural network training and prediction[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 1231-1247.

[110]. Mishra P, Lehmkuhl R, Srinivasan A, et al. Delphi: A cryptographic inference service for neural networks[C]//29th {USENIX} Security Symposium ({USENIX} Security 20). 2020: 2505-2522.

[111]. Rathee D, Rathee M, Kumar N, et al. CrypTFlow2: Practical 2-party secure inference[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 325-342.

[112]. Mohassel P, Rindal P. ABY^3: A mixed protocol framework for MAChine learning[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 35-52.

[113]. Chaudhari H, Rachuri R, Suresh A. Trident: Efficient 4PC framework for privacy preserving MAChine learning[C]//27th Annual Network and Distributed System Security Symposium, NDSS 2020. 2020.

[114]. Kumar N, Rathee M, Chandran N, et al. CrypTFlow: Secure tensorflow inference[C]//2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020: 336-353.

[115]. Huang Y, Evans D, Katz J. Private set intersection: Are garbled circuits better than custom protocols?[C]//NDSS. 2012.

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

[117]. Ciampi M, Orlandi C. Combining private set-intersection with secure two-party computation[C]//International Conference on Security and Cryptography for Networks. Springer, Cham, 2018: 464-482.

[118]. Dong C, Chen L, Wen Z. When private set intersection meets big data: an efficient and scalable protocol[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013: 789-800.

[119]. Pinkas B, Schneider T, Segev G, et al. Phasing: Private set intersection using permutation-based hashing[C]//24th {USENIX} Security Symposium ({USENIX} Security 15). 2015: 515-530.

[120]. Kolesnikov V, Matania N, Pinkas B, et al. Practical multi-party private set intersection from symmetric-key techniques[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1257-1272.

[121]. Rindal P, Rosulek M. Improved private set intersection against malicious adversaries[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2017: 235-259.

[122]. Melissa Chase, Peihan Miao: Private Set Intersection in the Internet Setting from Li ghtweight Oblivious PRF. CRYPTO (3) 2020: 34-63.

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

[124]. Chor, Benny, et al. "Private information retrieval." *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995.

[125]. H. Sun and S. A. Jafar, "The Capacity of Private Information Retrieval," in IEEE Transactions on Information Theory, vol. 63, no. 7, pp. 4075-4088, July 2017.

[126]. H. Sun and S. A. Jafar, "The Capacity of Robust Private Information Retrieval With Colluding Databases," in IEEE Transactions on Information Theory, vol. 64, no. 4, pp. 2361-2370, April 2018.

[127]. Kushilevitz, Eyal, and Rafail Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval." *Proceedings 38th annual symposium on foundations of computer science*. IEEE, 1997.

[128]. Cachin, Christian, Silvio Micali, and Markus Stadler. "Computationally private information retrieval with polylogarithmic communication." *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 1999.

[129]. Chor, Benny, and Niv Gilboa. "Computationally private information retrieval." *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 1997.

[130]. C. Gentry, “Fully Homomorphic Encryption Scheme,” PhD thesis, Stanford Univ., manuscript, http://crypto.stanford.edu/craig, 2009.

[131]. X. Yi, M. G. Kaosar, R. Paulet and E. Bertino, "Single-Database Private Information Retrieval from Fully Homomorphic Encryption," in IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 5, pp. 1125-1134, May 2013.

[132]. Angel S, Chen H, Laine K, et al. PIR with compressed queries and amortized query processing[C]//2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 962-979.

[133]. Melchor C A, Barrier J, Fousse L, et al. XPIR: Private information retrieval for everyone[J]. Proceedings on Privacy Enhancing Technologies, 2016, 2016: 155-174.

[134]. Hastings M, Hemenway B, Noble D, et al. Sok: General purpose compilers for secure multi-party computation[C]//2019 IEEE symposium on security and privacy (SP). IEEE, 2019: 1220-1237.

[135]. Malkhi D, Nisan N, Pinkas B, et al. Fairplay-Secure Two-Party Computation System[C]//USENIX Security Symposium. 2004, 4: 9.

[136]. Wang X, Malozemoff A J, Katz J. EMP-toolkit: Efficient MultiParty computation toolkit[J]. 2016.

[137]. Zahur S, Evans D. Obliv-C: A Language for Extensible Data-Oblivious Computation[J]. IACR Cryptol. ePrint Arch., 2015, 2015: 1153.

[138]. Liu C, Wang X S, Nayak K, et al. Oblivm: A programming framework for secure computation[C]//2015 IEEE Symposium on Security and Privacy. IEEE, 2015: 359-376.

[139]. Mood B, Gupta D, Carter H, et al. Frigate: A validated, extensible, and efficient compiler and interpreter for secure computation[C]//2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2016: 112-127.

[140]. Holzer A, Franz M, Katzenbeisser S, et al. Secure two-party computations in ANSI C[C]//Proceedings of the 2012 ACM conference on Computer and communications security. 2012: 772-783.

[141]. Franz M, Holzer A, Katzenbeisser S, et al. CBMC-GC: an ANSI C compiler for secure two-party computations[C]//International Conference on Compiler Construction. Springer, Berlin, Heidelberg, 2014: 244-249.

[142]. Büscher N, Holzer A, Weber A, et al. Compiling low depth circuits for practical secure computation[C]//European Symposium on Research in Computer Security. Springer, Cham, 2016: 80-98.

[143]. Burra S S, Larraia E, Nielsen J B, et al. High Performance Multi-Party Computation for Binary Circuits Based on Oblivious Transfer[J]. IACR Cryptol. ePrint Arch., 2015, 2015: 472.

[144]. Chou T, Orlandi C. The simplest protocol for oblivious transfer[C]//International Conference on Cryptology and Information Security in Latin America. Springer, Cham, 2015: 40-58.

[145]. Mansy D, Rindal P. Endemic oblivious transfer[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 309-326.

[146]. Peikert C, Vaikuntanathan V, Waters B. A framework for efficient and composable oblivious transfer[C]//Annual international cryptology conference. Springer, Berlin, Heidelberg, 2008: 554-571.

[147]. Asharov G, Lindell Y, Schneider T, et al. More efficient oblivious transfer and extensions for faster secure computation[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013: 535-548.

[148]. Rastogi A, Hammer M A, Hicks M. Wysteria: A programming language for generic, mixed-mode multiparty computations[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 655-670.

[149]. Zhang Y, Steele A, Blanton M. PICCO: a general-purpose compiler for private distributed computation[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013: 813-826.

[150]. Demmler D, Schneider T, Zohner M. ABY-A framework for efficient mixed-protocol secure two-party computation[C]//NDSS. 2015.

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

[152]. Aly A, Orsini E, Rotaru D, et al. Zaphod: Efficiently combining LSSS and garbled circuits in SCALE[C]//Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography. 2019: 33-44.

[153]. Keller M. MP-SPDZ: A versatile framework for multi-party computation[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 1575-1590.

[154]. Büscher N, Demmler D, Katzenbeisser S, et al. HyCC: Compilation of hybrid protocols for practical secure computation[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 847-861.

[155]. Dai W, Sunar B. cuHE: A homomorphic encryption accelerator library[C]//International Conference on Cryptography and Information Security in the Balkans. Springer, Cham, 2015: 169-186.

[156]. Dathathri R, Saarikivi O, Chen H, et al. CHET: an optimizing compiler for fully-homomorphic neural-network inferencing[C]//Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2019: 142-156.

[157]. Boemer F, Lao Y, Cammarota R, et al. ngraph-he: A graph compiler for deep learning on homomorphically encrypted data[C]//Proceedings of the 16th ACM International Conference on Computing Frontiers. 2019: 3-13.

[158]. Brandão L T A N, Mouha N, Vassilev A. Threshold schemes for cryptographic primitives: challenges and opportunities in standardization and validation of threshold cryptography[R]. National Institute of Standards and Technology, 2018.

[159]. Brandão L T A N, Davidson M, Vassilev A. NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives[R]. National Institute of Standards and Technology, 2020.

[160]. ISO/IEC 18033-6:2019 .IT Security techniques — Encryption algorithms — Part 6: Homomorphic encryption.https://www.iso.org/standard/67740.html.

[161]. Albrecht M R, Chase M, Chen H, et al. Homomorphic Encryption Standard[J]. IACR Cryptol. ePrint Arch., 2019, 2019: 939.

[162]. Bos J W, Lauter K, Loftus J, et al. Improved security for a ring-based fully homomorphic encryption scheme[C]//IMA International Conference on Cryptography and Coding. Springer, Berlin, Heidelberg, 2013: 45-64.

[163]. Hoffstein J, Pipher J, Silverman J H. NTRU: A ring-based public key cryptosystem[C]//International Algorithmic Number Theory Symposium. Springer, Berlin, Heidelberg, 1998: 267-288.

[164]. Chase M, Chen H, Ding J, et al. Security of homomorphic encryption[J]. HomomorphicEncryption. org, Redmond WA, Tech. Rep, 2017.

[165]. Brenner M, Dai W, Halevi S, et al. A standard api for rlwe-based homomorphic encryption[R]. Technical report, HomomorphicEncryption. org, Redmond WA, 2017.

[166]. Archer D, Chen L, Cheon J H, et al. Applications of homomorphic encryption[J]. HomomorphicEncryption. org, Redmond WA, Tech. Rep., 2017.