计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200400091

MASCOT 协议的参与方自适应变体

李艳斌[1]　刘　瑜[2]　李木舟[3]　吴韧韬[1]　王鹏达[1]
1 电科云(北京)科技有限公司　北京 100041
2 潍坊学院计算机工程学院　山东 潍坊 261061
3 山东大学(青岛校区)网络空间安全学院　山东 青岛 266237
(liyanbin@cetccloud.com)

摘　要　在过去十年中,安全多方计算(secure Multi-Party Computation,MPC)已经从纯理论研究发展到成为构建隐私保护应用程序的重要多功能工具。在 CCS 2016 上,Keller 等提出安全多方计算协议-MASCOT,其预处理阶段基于不经意传输协议,而不是类似经典 SPDZ 协议采用的部分同态加密技术。这使得 MASCOT 的性能相比 SPDZ 提升了两个数量级。由于其出色性能和高可用性,MASCOT 引起了工业界的广泛关注。但在实际应用环境中,仍然存在 MASCOT 不能满足的用户需求。其中主要的缺点是 MASCOT 无法支持在线计算阶段中发生的参与方变更。一个直观的解决方式是在对新的参与方集合重新运行预处理阶段,重新生成在线计算所需的数据材料。但是这明显造成了数据资源与时间的浪费。针对这一实际应用需求,文中在 MASCOT 的主要组件中进行技术微调,使其适应各类参与方集合发生变化的情况,包括新参与方加入、旧参与方退出以及新参与方替代旧参与方。将对预处理数据材料的处理限制在发生变更的参与方之间,或发生变更的参与方与未发生变更的参与方之间,避免在参与方集合中重新执行整个预处理阶段,有效降低适应参与方变更所需的数据与时间资源。此外,对 MASCOT 的微调是在保证与原 MASCOT 一致的功能、性能与安全性的前提下进行的。因此,MASCOT 的参与方自适应变体更接近实际应用环境,适合广泛配置在隐私保护应用程序中。对已经配置了 MASCOT 协议的应用程序,也能快速地采用所提技术添加参与方自适应性。

关键词:安全多方计算;SPDZ;MASCOT;参与方自适应;隐私保护;乘法三元组;可认证加法分片
中图法分类号　TP309.2

Participant-adaptive Variant of MASCOT

LI Yan-bin[1],LIU Yu[2],LI Mu-zhou[3],WU Ren-tao[1] and WANG Peng-da[1]
1 CETC Cloud (Beijing) Technology Co.,LTD,Beijing 100041,China
2 School of Computer Engineering,Weifang University,Weifang,Shangdong 261061,China
3 School of Cyber Science and Technology,Shandong University (Qingdao Campus),Qingdao,Shangdong 266237,China

Abstract　Over the last decade,secure multi-party computation (MPC) has made a great stride from a major theoretical area to the multi-functional tool for building privacy-preserving applications. At CCS 2016,Keller et al. presented MPC protocol MASCOT with preprocessing phase based on oblivious transfer (OT),instead of somewhat homomorphic encryption that classical SPDZ adopts,which improves by two orders of magnitude compared to SPDZ. Due to its superior performance and high availability,MASCOT has drew a lot of attention from industry. But in practical application environment,there are still users' needs that MASCOT cannot satisfy. The main disadvantage is that it is unable to handle changes in the set of parties during online computing phase. A straight forward solution is to regenerate the raw data materials required for online computation by rerunning the entire preprocessing phase among the new set of parties,which obviously results in a serious waste of data and time resources. For this practical issue,the main components of MASCOT are tweaked to adapt to the various changes of the set of parties,including new parties joining in,old parties dropping out and new parties replacing old parties. By strictly restricting the communications for pre-processed data to parties that have changed,or between parties those have changed and who have not changed,the whole preprocessing phase is avoided to be redone among parties remained after change,and it effectively reduces the data and time for suit parties changing. In addition,the minor modification of MASCOT is carried out on the premise of ensuring the functionality,performance and security consistent with the original MASCOT. In a word,the participant-adaptive variant of MASCOT

本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:全国一体化国家大数据中心先导工程(X06002019004);国家自然科学基金(61902283);潍坊学院 2019 年博士科研启动基金(2019BS13)
This work was supported by the Integration of National Big Data Center Pilot Project (X06002019004),National Natural Science Foundation of China (61902283) and 2019 Phd Research Start-up Fund of Weifang University (2019BS13).
通信作者:刘瑜(yuliu@wfu.edu.cn)

is closer to the actual application environment and is suitable for extensive deployment in applications with privacy. The technique can also be easily used to add participant adaptability to deployed MASCOT protocol as it only fine-tunes the preprocessing phase in a subtle way.

Keywords Multi-party computation, SPDZ, MASCOT, Participant-adaptive, Privacy-preserving, Multiplication triple, Authenticated additive sharing

1 引言

安全多方计算[1]是理论与应用密码学的研究热点，它解决了一组互不信任但又希望保护本身隐私数据的各参与方之间的协作计算的问题。简而言之，在 n 个参与方 $P_1, P_2, \cdots, P_n$ 中，各自持有私有数据 $x_1, x_2, \cdots, x_n$，其目标是合作计算公开给定的函数 $f(x_1, x_2, \cdots, x_n)$。在这一过程中，MPC 协议必须保证输入的机密性和独立性，以及计算结果的正确性。安全的 MPC 协议即使在一部分参与方是不诚实的情况下，仍然具有让诚实的参与方获得协议唯一发布的新信息的能力，这个新信息是正确计算结果。近十年来，出于对隐私保护的需求，基于 MPC 的应用程序出现得越来越多。例如，MPC 在为电子拍卖[2-4]、电子投票[5-6]、统计分析[7]和机器学习[8]提供机密性方面发挥了重要作用。除此之外，很多区块链[9-10]应用也配置了 MPC，增加了保护隐私的功能[11-12]。

大多数 MPC 协议都可以实现函数的求值，协议中将函数看作有限域上的布尔或算术电路。Yao 的混淆电路协议首次实现了 MPC，虽然它只能用于计算两方之间的布尔电路[13]。随后，越来越多的 MPC 协议出现，其中重要的有 GMW[14-15]、BGW[16]、BMR[17]、GESS[18-19]、BDOZ[20]、SPDZ[21]等。SPDZ 的设计相比其他协议是非常成熟的。针对 SPDZ，又涌现出了很多优化版本。原 SPDZ 协议采用 Schnorr-like 协议[22]实现零知识（Zero-Knowledge，ZK）证明，从而保证各参与方确实加密了应该加密的信息。文献[23]使用 cut-and-choose 技术代替了 Schnorr-like 协议，解决了许多文献[21]未解决的问题。为了在它们之间进行区分，这两个协议分别被称为 SPDZ-1[21]和 SPDZ-2[23]。SPDZ-1 的设计支持预处理模型，在获知待计算的函数与输入之前生成必须的数据材料，使得在线计算阶段只采用廉价的信息理论元，从而大幅度提高效率。因此，这种模式被广泛应用在随后的 MPC 协议中。SPDZ-1 与 SPDZ-2 都采用了部分同态加密技术（Somewhat Homomorphic Encryption，SHE）作为预处理阶段的主要技术支撑。在 CCS 2016 上，Keller 等提出了 MASCOT[24]，其预处理阶段基于不经意传输协议（oblivious transfer，OT）。与 SPDZ-2 相比，MASCOT 将性能提升了 2 个数量级。一个进一步的提升出现在 EUROCRYPT 2018，SPDZ-Overdrive[25]的实现表明基于 SHE 的预处理阶段的处理速度更快，它采用的 SHE 是仅对加法同态的半同态加密（semi-Homomorphic Encryption，semi-HE），而不是支持深度为 1 的乘法的 SHE[21,23]。但在这篇论文中，我们关注于 MASCOT，在一次性设置阶段后，可以完全基于对称密码元的 MPC 协议。

目前多个实现 MPC 协议的框架已经在学术领域出现，例如 ABY^3[26]，CBMC-GC[27]，EMP-toolkit[28]，FRESCO[29]，Frigate[30]，JIFF[31]，MPyC[32]，Obliv-C[33]，OblivVM[34]，PICCO[35]，SCALE-MAMBA[36]，MP-SPDZ[37]等。这使得工业界将 MPC 协议应用到需求隐私保护的应用程序成为可能，从而在实际环境中响应当前数据安全方面的法律和法规，比如欧盟在 2018 年引入的《通用数据保护条例》（General Data Protection Regulation，GDPR），中国在 2017 年实施的《中华人民共和国网络安全法》和《中华人民共和国民法总则》都表明对数据隐私和安全管理日趋严格。这些法规的建立在不同程度上对传统数据处理模式提出了新的挑战。MPC 的实质是为多领域多机构的数据共享交换打造一个“数据可用不可见”的安全共享环境，打破当前出于数据安全性的考虑而存在的数据壁垒和信息孤岛。但在实际应用程序中添加 MPC，仍然存在很多实际用户需求无法满足，其中最主要的是大多数 MPC 协议作为理论研究成果，无法处理在线计算阶段参与方出现变更的情况。当然，最直接的解决办法是在新的参与方集合中重新运行整个预处理阶段，抛弃之前生成的所有预处理数据和计算中间结果。很明显这一解决办法造成了大量数据和时间资源的浪费。

在线计算阶段参与方发生变更的实际情况主要包括新参与方加入和旧参与方退出。为了以最小成本处理这一问题，在不损害安全性的前提下，我们对 MASCOT 进行微调，对已经存在的预处理数据和部分中间计算结果进行妥善处理，以适合新的参与方集合。本文的主要贡献如下：

(1)通过对 MASCOT 内部技术细节的仔细观察，我们增加或减少了已变更参与方之间或已变更参与方与未变更参与方之间的必要交互。这里的时间成本只是为了填补由参与方变更引起的数据格式差异，而无需重新执行预处理。此外，在微调过程中严格遵循原 MASCOT 对安全性的处理原则，保持与其相同的安全级别。

(2)考虑到在应用程序中部署协议的工作量，协议使用时间越长，其中优化的组件就越多，后续修改的代价就越大。因此寻找最小的协议调整以达到最大功能、性能优化或更高的安全性是非常重要的。我们对 MASCOT 的微调符合这一准则，它以较小修改解决了 MASCOT 的一个实际应用问题。

文中使用的符号如表 1 所列。

表 1 符号表示

Table 1 Notations and operations

$\mathbb{F}$	有限域 $\mathbb{F}=\mathbb{F}_p$ 或 $\mathbb{F}=\mathbb{F}_{2^k}$
$\mathscr{C}$	有限域 $\mathbb{F}$ 上待计算的电路
x	有限域 $\mathbb{F}$ 中的值
$x \xleftarrow{\$} \mathbb{F}$	随机抽取有限域 $\mathbb{F}$ 中的值并赋给 x
$\boldsymbol{x}$	有限域 $\mathbb{F}$ 上的向量
$[\![x]\!], x^i$	x 的可认证加法分片方案，P_i 持有 x 的分片
$\Delta, \Delta^{(i)}$	秘密 MAC 密钥及它被 P_i 持有的分片
$m^{(i)}, m(x)^{(i)}$	P_i 持有的 x 的 MAC 分片
P_{old}, n	参与方发生变更前，n 个参与方的集合
P_{new}, N	参与方发生变更后，N 个参与方的集合
Enc	SHE 或 semi-HE 的加密函数
Dec	SHE 或 semi-HE 的分布式解密协议
τ	影响 MASCOT 安全性的常量
$g(\cdot), g^{-1}(\cdot)$	工具向量与比特分解函数

本文第 2 节给出了文中使用的符号表示；第 3 节简要介

绍了 SPDZ 与 MASCOT 协议的通用设计；第 4 节对 MASCOT 的预处理阶段进行了微调，并在第 5 节中给出了 MASCOT 的参与方自适应变体协议；最后总结全文。

2 SPDZ 与 MASCOT

SPDZ[21,23]在互不信任的参与方之间高效地完成隐私保护任务，它包含两个阶段：预处理阶段与在线阶段。预处理阶段的主要工作是生成随机值与乘法三元组的可认证加法分片方案，其中涉及的数据独立于待计算的电路与参与方的输入。这一阶段可以提取完成，为在线阶段做准备。在线阶段则在接收参与方输入后执行电路计算。自从 SPDZ 发布以来，有许多优化协议出现，其中预处理阶段要么基于 SHE，要么基于 OT。其中，MASCOT 是唯一基于 OT 协议的优化。本节首先介绍了 SPDZ 协议及其优化协议的架构，并给出了 MASCOT 协议的简要描述。

2.1 在线阶段

为提供涵盖机密性与可认证性的积极安全性，有限域 $\mathbb{F}$ 上的秘密值 x 的保护是将其在 $\mathbb{F}$ 上进行加法秘密分割并结合信息理论 MAC。即：

$$[\![x]\!]=(x^{(1)},\cdots,x^{(n)},m^{(1)},\cdots,m^{(n)},\Delta^{(1)},\cdots,\Delta^{(n)})$$

其中，参与方 P_i 持有随机分片 $x^{(i)}$、随机 MAC 分片 $m^{(i)}$ 以及固定的 MAC 密钥分片 $\Delta^{(i)}$，并在 $\mathbb{F}$ 上有：

$$x=\sum_{i=1}^{n}x^{(i)},\ m=\sum_{i=1}^{n}m^{(i)},\ \Delta=\sum_{i=1}^{n}\Delta^{(i)}$$

同时成立 MAC 关系式：$m=x\cdot\Delta$。

如果需要打开秘密值 $[\![x]\!]$，所有参与方 $P_i(1\leqslant i\leqslant n)$ 广播它们各自持有的分片 $x^{(i)}$ 并计算 x，然后提交并打开 $m^{(i)}-x\cdot\Delta^{(i)}$。通过检测这些分片的和是否为 0，来验证已经代开的 x 的值的有效性。取多个秘密值的分片与 MAC 分片的随机线形组合可以一次性验证多个秘密值，具体细节见协议 1。

在线阶段旨在对有限域上以参与方私有数据为输入的任意电路求值。在各方确定了各自的输入后，有必要为其快速构建可认证分片方案。这可以通过各方在预处理阶段生成大量随机的可认证共享值。即 P_i 随机选取 r，与其他各方合作生成 $[\![r]\!]$。则对 P_i 在线阶段的私有输入 x，广播 $\sigma=x-r$，各方通过计算 $[\![x]\!]=[\![r]\!]+\sigma$ 对 x 进行认证和共享。当计算电路时，为了图灵完备性，在可认证加法共享值上重定义加法与乘法的运算规则是非常重要的。由于共享方案是基于加法的，可认证共享值计算加法只需各方执行简单的本地加法计算而无需交互，即 $[\![x+y]\!]=[\![x]\!]+[\![y]\!]$。乘法较为复杂，在计算 $[\![x\cdot y]\!]$ 时需要各方通信。当前最为有效的方法是利用 Beaver's trick[38]，在生成乘法三元组 $([\![a]\!],[\![b]\!],[\![c]\!])(c=a\cdot b)$ 的基础上，$[\![x]\!]$ 与 $[\![y]\!]$ 的可认证加法共享的乘积为：

$$[\![x\cdot y]\!]=[\![c]\!]+\varepsilon\cdot[\![b]\!]+\rho\cdot[\![a]\!]+\varepsilon\cdot\rho,$$

其中，ε 和 ρ 分别是 $[\![x-a]\!]$ 与 $[\![y-b]\!]$ 的打开值。

2.2 预处理阶段

从在线阶段的描述中可以看出，预处理阶段的主要任务是生成以下两种随机的、可认证的共享值。

(1) P_i 的随机输入：$([\![r]\!],i)$，对于随机的 r 值，只有 P_i 知道；

(2) 乘法三元组：$([\![a]\!],[\![b]\!],[\![c]\!])$，对于随机的 a 和 b，有 $c=a\cdot b$。

此外，由于待计算电路的多样性，平方组 $([\![a]\!],[\![a^2]\!])$、逆元组 $([\![a]\!],[\![a^{-1}]\!])$ 也可以提前准备。在这些元组中，乘法三元组的生成是最重要的组件，因为其他元组是基于乘法三元组生成的。当 Δ 是秘密共享的全局随机 MAC 密钥时，乘法三元组本质上是生成 $(a,b,a\cdot b,a\cdot\Delta,b\cdot\Delta,a\cdot b\cdot\Delta)$ 的加法分片方案，其中 a 和 b 是随机的。各方在本地选择相应随机的分片，得到 a,b 和 Δ 是很容易的。真正棘手的部分是生成剩余乘积的秘密分片方案。

基于 SHE 的预处理阶段需要与 Δ 对应的公钥 pk 和分布式解密协议，允许参与方在不知道 Δ 的情况下解密密文，并收到相应明文的一个分片。在仅提供消极安全性的前提下，各方随机选择自己的分片 $a^{(i)},b^{(i)}$ 和 MAC 密钥分片 $\Delta^{(i)}$。然后他们利用公钥 pk 对 $a^{(i)},b^{(i)}$ 和 $\Delta^{(i)}$ 进行加密，并广播 $Enc_{pk}(a^{(i)}),Enc_{pk}(b^{(i)})$ 和 $Enc_{pk}(\Delta^{(i)})$。由于 Enc 的同态特性，$(a\cdot b,a\cdot\Delta,b\cdot\Delta,a\cdot b\cdot\Delta)$ 的密文可以通过对 $Enc_{pk}(a^{(i)}),Enc_{pk}(b^{(i)})$ 和 $Enc_{pk}(\Delta^{(i)})$ 做加法和乘法得到。如果 Enc 支持的乘法深度为 2，则在所有参与方之间运行分布式解密协议 Dec，能成功获得乘积的有效加法分片方案。

这种 SHE 用于三元组生成明显是冗余的。事实上，支持乘法深度为 1 的 SHE 足够处理 $a\cdot b$ 的分片方案，以及随后 $a\cdot b\cdot\Delta$ 的分片方案。SPDZ 采用了这样的设计，而在 SPDZ-Overdrive[25]中进一步降低了对 SHE 的需求，它只需支持加法的 semi-HE。一方 P_i 将其输入 a 的密文 $Enc_{pk}(a)$ 发送给另一方 P_j，P_j 选取输入 b 和随机数 c_B，计算 $C\leftarrow b\cdot Enc_{pk}(a)-Enc_{pk}(c_B)$ 并将 C 回复给 P_i。因为 semi-HE 的加密函数 Enc 允许已知值与密文之间的乘法，P_i 解密 C 得到明文 $c_A\leftarrow b\cdot a-c_B$，这就得到了 $a\cdot b$ 的一个加法共享方案 (c_A,c_B)。在每两个参与方之间执行上述过程，乘法三元组就能高效地产生。这种交互方式与 MASCOT[24]非常相似，不同的是 MASCOT 中交互的信息采用了 OT 技术而不是 SHE。即在基于 OT 的预处理阶段中，在每对参与方之间运行 OT 实例来创建乘法三元组。随机输入元组则是在每两方之间运行相关不经意乘积分解(correlated oblivious product evaluation, COPE)协议，其中 COPE 协议是消极安全的 OT 扩展协议[39]的算术推广，允许两个参与方对乘积 $x\cdot\Delta$(一方持有 x，另一方持有 Δ)进行加法分解。

为进一步提供积极安全性，乘法三元组的正确性与私有性需要更多的技术手段来消除生成过程中的错误带来的影响。对基于 SHE 的预处理阶段，分布式解密可能会引入错误，导致三元组 $([\![a]\!],[\![b]\!],[\![ab+e]\!])$ 无法通过 MAC 检测。此时，该三元组的正确性可以“牺牲”另一个三元组 $([\![a']\!],[\![b']\!],[\![a'b'+e']\!])$ 来检查。选择一个随机值 t，正确性检查的过程是对 $t\cdot(ab+e)-(a'b'+e')-(ta-a')\cdot b-a'\cdot(b-b')$ 求值，当该式为 0 时忽略错误三元组的概率可忽略不计。在 MASCOT 中，正确性的验证方式是类似的，采用具有相同 b 值的一对三元组 $([\![a]\!],[\![b]\!],[\![ab]\!])$ 和 $([\![a']\!],[\![b]\!],[\![a'b]\!])$。而 SPDZ-Overdrive 则采用具有相同 a 值的一对三元组 $([\![a]\!],[\![b]\!],[\![ab]\!])$ 和 $([\![a]\!],[\![b']\!],[\![ab']\!])$。此外，ZK 广泛应用于基于 SHE 的预处理阶段，确保各方确实加密了他们应该加密的数据。基于 OT 的预处理阶段利用隐私放大技术从多个已泄漏部分比特的三元组中提取随机三元组。例如，对于相关向量三元

组($\boldsymbol{a},b,\boldsymbol{c}$)，在 $\boldsymbol{a}$ 或 $\boldsymbol{c}$ 与随机向量 $\boldsymbol{r}$ 之间取内积，从而获得新的三元组(a,b,c)，其中 $a=\langle\boldsymbol{a},\boldsymbol{r}\rangle$，$c=\langle\boldsymbol{c},\boldsymbol{r}\rangle$。这样在 $\boldsymbol{a}$ 和 $\boldsymbol{c}$ 中任意泄露的比特会与没有泄露的比特一起随机化。

2.3 MASCOT

MASCOT 协议作为 SPDZ 协议的优化之一，其架构与上文中的描述一致。这里主要给出 MASCOT 协议中的某些技术细节，为后文中的工作做准备。

如图 1 所示，在 MASCOT 协议的预处理阶段，OT 协议扮演了十分重要的部分。OT 协议需要一方的输入是选择比特，因此在基于 OT 协议的 Π_{COPEe} 中，要求输入是以比特向量的形式，为了将域元素转换为比特向量，MASCOT 采用了与格密码学中常用的符号——工具向量 $\boldsymbol{g}$，其分量为 2(有限域 $\mathbb{F}_p$)或 X(有限域 $\mathbb{F}_{2^k}$)，使得：

$$\boldsymbol{g}=(1,g,g^2,\cdots,g^{k-1})\in\mathbb{F}^k$$

其中，在 $\mathbb{F}_p$ 上 $g=2$，在 $\mathbb{F}_{2^k}$ 上 $g=X$。令 $\boldsymbol{g}^{-1}:\mathbb{F}\to\{0,1\}^k$ 表示将 $x\in\mathbb{F}$ 映射到比特向量 $\boldsymbol{x}_B=\boldsymbol{g}^{-1}(x)\in\{0,1\}^k$ 的“比特分解”函数，并且通过取内积 $\langle\boldsymbol{g},\boldsymbol{g}^{-1}(x)\rangle=x$ 可以将 $\boldsymbol{x}_B$ 映射回 $\mathbb{F}$。这个基本工具使得我们能在域元素与比特向量之间容易地转换，同时又不依赖于底层的有限域。

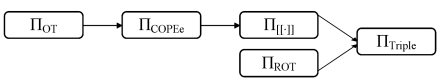

图 1 MASCOT 协议预处理阶段的组件依赖关系

Fig. 1 Dependency among components in preprocessing phase of MASCOT

为了保证足够的安全水平，Π_{Triple} 中首先生成三元组向量($\boldsymbol{a},b,\boldsymbol{c}$)的分片，其中 $\boldsymbol{a},\boldsymbol{c}\in\mathbb{F}^\tau$，而 τ 是影响 MASCOT 安全性的常量，一般来说，$\tau=3/4$ 就能确保积极安全性。其次以随机向量联合 $\boldsymbol{a}$ 或 $\boldsymbol{c}$ 取内积从而将 $\boldsymbol{a},\boldsymbol{c}$ 中可能出现泄露的比特与其他比特充分混淆。

协议 1 $\Pi_{MACCheck}$

在 P_i 中输入(MACCheck,N,x,$m(x)^{(i)}$,$\Delta^{(i)}$)，P_i 执行：

1. 计算 $\sigma^{(i)}\leftarrow m(x)^{(i)}-x\cdot\Delta^{(i)}$。
2. 调用 $\mathcal{F}_{Commit}$(Commit,$\sigma^{(i)}$)得到 τ_i。
3. 在(Open,τ_i)后调用 $\mathcal{F}_{Commit}$ 将 $\sigma^{(i)}$ 广播至所有参与方。
4. 如果 $\sigma^{(1)}+\cdots+\sigma^{(N)}\neq0$，中止并输出 $\perp$；否则继续。

3 微调预处理阶段

当参与方集合在在线阶段要求变更时，令 P_{old} 表示原包含 n 个参与方的集合；而 P_{new} 表示变更发生后包含 N 个参与方的集合。一般来说对 P_{new} 有 3 种情况。当有新参与方想要加入计算，即 $P_{new}\supset P_{old}$ 时，有必要对已经生成的预处理数据的可认证分片方案进行扩展，从而适应随后在 P_{new} 上发生的计算。当原参与方想要退出计算，即 $P_{new}\subset P_{old}$ 时，这些退出方持有的分片需要妥善处理，避免数据泄露，同时对已存在的预处理数据的可认证分片方案进行降维处理。较为复杂的情况是原参与方被某些新的参与方替代了，解决方案可以分为两个步骤：1)原参与方退出；2)新参与方加入。因此，通过运行前两种情况对应的解决方案能成功地解决该问题，这里我们不再对这种情况进行详细描述。

预处理阶段主要包括两个组件，分别用于生成两种类型的随机的可认证共享值。我们给出在这些组件上进行的微调技术，基于这些技术构建具有参与方自适应性的 MASCOT 协议的变体。

3.1 $\boldsymbol{\Pi}_{[\![\cdot]\!]}$

$\Pi_{[\![\cdot]\!]}$ 旨在为各方提交的输入数据及其线形组合的秘密可认证共享方案。值得注意的是，这里的输入不是在线阶段中电路需求的输入，而是各方随机选取的预处理阶段的输入值，对这些随机值进行分片处理有助于在在线阶段高效地为电路输入提供可认证共享方案。$\Pi_{[\![\cdot]\!]}$ 包含 5 个子组件，我们将对它们以及对它们引入的微调逐一进行介绍。

Initialize 用于为参与方集合 P_{old} 设置 MAC 密钥分片，并在每对参与方中执行 COPE 协议的初始化。Initialize 只需执行一次。

Initialize 具体如下。

输入(Initialize,P_{old})，各方 P_i($1\leqslant i\leqslant n$)执行：

1. 随机抽取一个 MAC 密钥分片 $\Delta^{(i)}\in\mathbb{F}$。
2. 对其他各参与方 P_j($1\leqslant j\leqslant n, j\neq i$)，调用 Π_{COPE}. Initialize($\mathbb{F}$)，其中 P_j 输入 $\Delta^{(j)}$。

当不属于 P_{old} 中的新参与方要求加入计算时，首先需要为其设置 MAC 密钥分片。其次需要在新参与方之间、新参与方与原参与方之间调用 Π_{COPEe}. *Initialize*($\mathbb{F}$)，为后续的随机输入处理做准备。

Initialize-nU 具体如下。

输入(Initialize,P_{new})，各方 P_i($n<i\leqslant N$)执行：

1. 随机抽取一个 MAC 密钥分片 $\Delta^{(i)}\in\mathbb{F}$。
2. 对于其他各参与方 P_j($1\leqslant j\leqslant N, j\neq i$)，调用 Π_{COPEe}. Initialize($\mathbb{F}$)，其中 P_i 输入 $\Delta^{(j)}$。

当 P_{old} 中某些参与方退出计算时，他们所持有的 MAC 密钥分片需要发送给仍然处于计算中的某个参与方，这里以 P_1 为例，以避免信息泄露。

Initialize-nD 具体如下。

输入(Initialize,P_{new})，各方 P_i($N<i\leqslant n$)执行：

1. 将 $\Delta^{(i)}$ 发送给 P_1。
2. P_1 更新其 MAC 密钥分片为 $\Delta^{(1)}=\Delta^{(1)}+\sum_{i=N+1}^{n}\Delta^{(i)}$。

在 MAC 密钥设置完成后，与其他各方之间调用协议对它进行处理，与其他各方之间分别调用 COPE 协议，从而得到输入值可认证加法分片方案。

Input P_j 具体如下。

输入：(Input,P_{old},id_1,$\cdots$,id_l,x_1,$\cdots$,x_l,P_j)，其他方输入(Input,P_{old},id_1,$\cdots$,id_l,P_j)

1. P_j 抽取 $x_0\xleftarrow{\$}\mathbb{F}$。
2. 对 $h=0,\cdots,l$，P_j 生成随机加法分片方案 $\sum_{i=1}^{n}x_h^{(i)}=x_h$，并发送 $x_h^{(i)}$ 到 P_i。
3. 对每一个 $i\neq j$，P_i 与 P_j 调用 Π_{COPEe}. Extend，其中 P_j 输入($x_0,x_1,\cdots,x_l$)$\in\mathbb{F}^{l+1}$。
4. 对 $h=0,\cdots,l$，P_i 接收 $q_h^{(i,j)}$，P_j 接收 $t_h^{(j,i)}$ 使得 $q_h^{(i,j)}+t_h^{(j,i)}=x_h\cdot\Delta^{(i)}$。
5. 对 P_i，$i\neq j$，定义其 MAC 分片 $m_h^{(i)}=q_h^{(i,j)}$，而 P_j 计算 MAC 分片为 $m_h^{(j)}=x_h\cdot\Delta^{(j)}+\sum_{j\neq i}^{t}t_h^{(j,i)}$，这样就得到了 $[\![x_h]\!]$($h=0,\cdots,l$)。
6. 各方抽取 $\mathbf{r}\leftarrow\mathcal{F}_{rand}(\mathbb{F}^{l+1})$。
7. P_j 计算并广播 $y=\sum_{h=0}^{l}r_h\cdot x_h$。

8. 各方P_i计算$m^{(i)}=\sum_{h=0}^{l} r_h \cdot m_h^{(i)}$。

9. 各方以 y 与$m^{(i)}$($1\leqslant i\leqslant n$)为输入调用$\Pi_{MACCheck}$。

10. 各方在标识符$id_1,\cdots,id_l$下存储相应的分片与 MAC 分片。

当新的参与方加入在线计算时,原参与方已经生成的可认证随机共享输入值必须重新设定在P_{new}中各方对应的分片与 MAC 分片,从而在在线阶段适应发生在P_{new}上的计算。而在参与方变更后提出的输入值的可认证共享方案可以直接以P_{new}为输入调用 Input 即可。

Input-nUP_j($1\leqslant i\leqslant n$)具体如下。

输入(Input,P_{new},$id_1,\cdots,id_l$,$x_1,\cdots,x_l$,P_j),其他各方P_i($1\leqslant i\leqslant N$,$i\neq j$)

输入(Input,P_{new},$id_1,\cdots,id_l$,P_j):

1. 对 $h=1,\cdots,l$,P_j重新生成随机加法分片方案$\sum_{i=1}^{N} x_h^{(i)}=x_h$,并发送$x_h^{(i)}$到$P_i$。$P_i$在收到此分片后,更新($1\leqslant i\leqslant n$)或设置($n<i\leqslant N$)标识符$id_h$对应的分片。

2. 对 $n<i\leqslant N$,P_i与P_j调用$\Pi_{COPEe}.Extend$,其中P_j输入$(x_1,\cdots,x_l)\in\mathbb{F}^l$。

3. 对 $h=1,\cdots,l$ 与 $n<i\leqslant N$,P_i接收$q_h^{(i,j)}$,P_j接收$t_h^{(j,i)}$使得$q_h^{(i,j)}+t_h^{(j,i)}=x_h\cdot\Delta^{(i)}$。

4. 对P_i,$n<i\leqslant N$,定义其 MAC 分片$m_h^{(i)}=q_h^{(i,j)}$,而P_j更新 MAC 分片为$m_h^{(j)}=m_h^{(j)}+\sum_{n<i\leqslant N} t_h^{(j,i)}$,这样就得到了$[\![x_h]\!]$($h=1,\cdots,l$)。

这里与原 Input 相比增加了 $N-n$ 次对$\Pi_{COPEe}.Extend$的调用,用以调整P_j的 MAC 分片和设置新加入的参与方持有的 MAC 分片。

当P_{old}中某些参与方要求退出计算时,他们所持有的已经生成的可认证共享值的分片与 MAC 分片必须发送给尚处于计算中的参与方。

Input-nD P_j($1\leqslant i\leqslant n$)具体如下。

输入(Input,P_{new},$id_1,\cdots,id_l$,$x_1,\cdots,x_l$,P_j),其他各方P_i($1\leqslant i\leqslant N$,$i\neq j$),输入(Input,P_{new},$id_1,\cdots,id_l$,P_j),P_k($N<k\leqslant n$)执行:

1. 对于 $h=1,\cdots,l$,将标识符id_h下存储的分片$x_h^{(i)}$与 MAC 分片$m_h^{(i)}$发送给P_1。

2. P_1更新id_h对应的分片为$x_h^{(1)}=x_h^{(1)}+\sum_{i=N+1}^{n} x_h^{(i)}$,MAC 分片为$m_h^{(1)}=m_h^{(1)}+\sum_{i=N+1}^{n} m_h^{(i)}$。

余下的 3 个子组件是在可认证共享值上进行的运算,包括线形组合、打开和 MAC 检测。在成功构建P_{new}上的可认证共享值后,这些运算与原P_{old}上的运算并无差别,我们在这里列出在P_{new}上的运算,以给出$\Pi_{[\![\cdot]\!]}$的总体描述。

LinearComb 具体如下。

输入(LinComb,P_{new},$\overline{id}$,$id_1,\cdots,id_t$,$c_1,\cdots,c_t$,c),各方提取在标识符$id_1,\cdots,id_t$下存储的分片与 MAC 分片$\{x_j^{(i)},m(x_j)^{(i)}\}_{1\leqslant j\leqslant t,1\leqslant i\leqslant N}$,各方$P_i$计算:

$$y^{(i)}=\sum_{j=1}^{t} c_j\cdot x_j^{(i)}+\begin{cases}c,i=1\\0,i\neq 1\end{cases}$$

$$m(y)^{(i)}=\sum_{j=1}^{t} c_j\cdot m(x_j)^{(i)}+c\cdot\Delta^{(i)}$$

之后在标识符$\overline{id}$下存储$[\![y]\!]$新的分片与 MAC 分片。

Open:

输入(Open,P_{new},id):

1. 各方P_i提取并广播 id 下存储的分片$x^{(i)}$。

2. 计算并输出$x=\sum_{i=1}^{N} x^{(i)}$。

Check:

输入(Check,P_{new},$id_1,\cdots,id_t$,$x_1,\cdots,x_t$),各方执行:

1. 抽取公开随机的向量$r\leftarrow\mathcal{F}_{rand}(\mathbb{F}^t)$。

2. 计算$y\leftarrow\sum_{j=1}^{t} r_j\cdot x_j$与$m(y)^{(i)}\leftarrow\sum_{j=1}^{t} r_j\cdot m_{id_j}^{(i)}$,其中$m_{id_j}^{(i)}$代表$P_i$存储在$id_j$下的 MAC 分片,$1\leqslant i\leqslant N$,$1\leqslant j\leqslant t$。

3. 以 y 与$m(y)^{(i)}$($1\leqslant i\leqslant N$)为输入调用$\Pi_{MACCheck}$。

3.2 Π_{Triple}

预处理阶段中另一个重要的组件是用于生成乘法三元组($[\![a]\!]$,$[\![b]\!]$,$[\![a\cdot b]\!]$)的Π_{Triple}。在Π_{Triple}中存在 4 个主要的子组件,我们对它们进行微调,增加参与方的自适应性。

生成三元组的($[\![a]\!]$,$[\![b]\!]$,$[\![a\cdot b]\!]$)的第一步是 Multiply,在各方随机选取的分片$a^{(i)}$与$b^{(i)}$的基础上,通过在每两个参与方之间运行$\Pi_{ROT}^{\tau k,k}$,生成乘积分片$(a\cdot b)^{(i)}$。

Multiply 具体如下。

输入(Multiply,P_{old}):

1. 各方抽取$a^{(i)}\xleftarrow{\$}\mathbb{F}^\tau$,$b^{(i)}\xleftarrow{\$}\mathbb{F}$。

2. 对于每对有序对(P_i,P_j):

2.1. 调用$\Pi_{ROT}^{\tau k,k}$,其中P_i输入$(a_1^{(i)},\cdots,a_{\tau k}^{(i)})=g^{-1}(a^{(i)})\in\mathbb{F}_2^{\tau k}$。

2.2. 对 $h=1,\cdots,\tau k$,P_j接收$q_{0,h}^{(j,i)},q_{1,h}^{(j,i)}\in\mathbb{F}$,$P_i$接收$s_h^{(i,j)}=q_{a_h^{(i)},h}^{(j)}$。

2.3. P_j发送$d_h^{(j,i)}=q_{0,h}^{(j,i)}-q_{1,h}^{(j,i)}+b^{(j)}$。

2.4. P_i设置$t_h^{(i,j)}=s_h^{(i,j)}+a^{(i)}\cdot d_h^{(j,i)}=q_{0,h}^{(j,i)}+a_h^{(i)}\cdot b^{(j)}$,令$q_h^{(j,i)}=q_{0,h}^{(j,i)}$。

2.5. 将$(t_1^{(i,j)},\cdots,t_{\tau k}^{(i,j)})$与$(q_1^{(j,i)},\cdots,q_{\tau k}^{(j,i)})$分别分割成为 τ 个 k 分量的向量$(\mathbf{t}_1,\cdots,\mathbf{t}_\tau)$与$(\mathbf{q}_1,\cdots,\mathbf{q}_\tau)$。

2.6. P_i设置$\mathbf{c}_{i,j}^{(i)}=(\langle\mathbf{g},\mathbf{t}_1\rangle,\cdots,\langle\mathbf{g},\mathbf{t}_\tau\rangle)\in\mathbb{F}^\tau$。

2.7. P_j设置$\mathbf{c}_{i,j}^{(j)}=-(\langle\mathbf{g},\mathbf{q}_1\rangle,\cdots,\langle\mathbf{g},\mathbf{q}_\tau\rangle)\in\mathbb{F}^\tau$。

2.8. 现在有$\mathbf{c}_{i,j}^{(i)}+\mathbf{c}_{i,j}^{(j)}=\mathbf{a}^{(i)}\cdot b^{(j)}\in\mathbb{F}^\tau$。

3. 各方P_i计算$\mathbf{c}^{(i)}=a^{(i)}\cdot b^{(i)}+\sum_{j\neq i}(\mathbf{c}_{i,j}^{(i)}+\mathbf{c}_{j,i}^{(i)})$。

当新参与方请求加入计算时,在已经生成的适用于参与方集合P_{old}的乘法三元组的基础上,通过在新参与方与新参与方以及新参与方与原参与方之间运行$\Pi_{ROT}^{\tau k,k}$,来获取新参与方所持有的三元组中的乘积分片$(a\cdot b)^{(i)}$。

Multiply-nU 具体如下。

输入(Multiply,P_{new}),各方P_i($n<i\leqslant N$)执行:

1. 抽取$\mathbf{a}^{(i)}\xleftarrow{\$}\mathbb{F}^\tau$,$b^{(i)}\xleftarrow{\$}\mathbb{F}$。

2. 对于每对有序对(P_i,P_j)($n<i\leqslant N$,$1\leqslant j\leqslant N$):

2.1. 调用$\Pi_{ROT}^{\tau k,k}$,其中P_i输入$(a_1^{(i)},\cdots,a_{\tau k}^{(i)})=\mathbf{g}^{-1}(\mathbf{a}^{(i)})\in\mathbb{F}_2^{\tau k}$。

2.2. 对 $h=1,\cdots,\tau k$,P_j接收$q_{0,h}^{(j,i)},q_{1,h}^{(j,i)}\in\mathbb{F}$,$P_i$接收$s_h^{(i,j)}=q_{a_h^{(i)},h}^{(j)}$。

2.3. P_j发送$d_h^{(j,i)}=q_{0,h}^{(j,i)}-q_{1,h}^{(j,i)}+b^{(j)}$。

2.4. P_i设置$t_h^{(i,j)}=s_h^{(i,j)}+a^{(i)}\cdot d_h^{(j,i)}=q_{0,h}^{(j,i)}+a_h^{(i)}\cdot b^{(j)}$,令$q_h^{(j,i)}=q_{0,h}^{(j,i)}$。

2.5. 将$(t_1^{(i,j)},\cdots,t_{\tau k}^{(i,j)})$与$(q_1^{(j,i)},\cdots,q_{\tau k}^{(j,i)})$分别分割成为 τ 个 k 分量的向量$(\mathbf{t}_1,\cdots,\mathbf{t}_\tau)$与$(\mathbf{q}_1,\cdots,\mathbf{q}_\tau)$。

2.6. P_i设置$\mathbf{c}_{i,j}^{(i)}=(\langle\mathbf{g},\mathbf{t}_1\rangle,\cdots,\langle\mathbf{g},\mathbf{t}_\tau\rangle)\in\mathbb{F}^\tau$。

2.7. P_j设置$\mathbf{c}_{i,j}^{(j)}=-(\langle\mathbf{g},\mathbf{q}_1\rangle,\cdots,\langle\mathbf{g},\mathbf{q}_\tau\rangle)\in\mathbb{F}^\tau$。

2.8. 现在有$\mathbf{c}_{i,j}^{(i)}+\mathbf{c}_{i,j}^{(j)}=\mathbf{a}^{(i)}\cdot b^{(j)}\in\mathbb{F}^\tau$。

3. P_i计算$\mathbf{c}^{(i)}=\mathbf{a}^{(i)}\cdot b^{(i)}+\sum_{j\neq i}(\mathbf{c}_{i,j}^{(i)}+\mathbf{c}_{j,i}^{(i)})$。

注意 Multiply-nU 与 Multiply 的步骤非常类似，区别是这些步骤仅需要对新加入的参与方执行，从而将P_{old}上的三元组升维以适用于P_{new}。具体来说升维操作消耗了N^2-n^2次$\Pi_{KOT}^{\mathbb{F}_{2^k}}$的调用，而重新运行整个过程生成新的三元组分片则需要N^2次$\Pi_{KOT}^{\mathbb{F}_{2^k}}$的调用。

随后，Combine 采用隐私放大技术，将可能泄露的比特与其他比特一起随机化，生成随机的三元组的分片。Authenticate 通过在随机分片上调用$\Pi_{[\![\cdot]\!]}$. *Input* 计算三元组相应的 MAC 分片。这样就可以成功地得到乘法三元组($[\![a]\!]$,$[\![b]\!]$,$[\![a\cdot b]\!]$)。而最后通过 Sacrifice 另一个三元组($[\![\hat{a}]\!]$,$[\![b]\!]$,$[\![\hat{a}\cdot b]\!]$)来检测($[\![a]\!]$,$[\![b]\!]$,$[\![a\cdot b]\!]$)中是否存在错误。这 3 步在技术细节上基本不受参与方变更的影响，我们在这里以P_{new}为参数列出这 3 步，从而给出Π_{Triple}的全部细节。

Combine：

输入(Multiply,P_{new})：

1. 抽取 $\mathbf{r},\hat{\mathbf{r}}\leftarrow\mathcal{F}_{\text{rand}}(\mathbb{F}^{\tau})$。

2. 各方P_i设置$a^{(i)}=\langle\mathbf{a}^{(i)},\mathbf{r}\rangle$，$c^{(i)}=\langle\mathbf{c}^{(i)},\mathbf{r}\rangle$以及$\hat{a}^{(i)}=\langle\mathbf{a}^{(i)},\hat{\mathbf{r}}\rangle$，$\hat{c}^{(i)}=\langle\mathbf{c}^{(i)},\hat{\mathbf{r}}\rangle$。

Authenticate：

输入(Authenticate,P_{new})：

1. 各方P_i在其分片上运行$\Pi_{[\![\cdot]\!]}$. Input，从而得到可认证共享值$[\![a]\!]$，$[\![b]\!]$，$[\![c]\!]$，$[\![\hat{a}]\!]$，$[\![\hat{c}]\!]$。

Sacrifice：

输入(Sacrifice,P_{new})，通过牺牲$[\![\hat{a}]\!]$，$[\![\hat{c}]\!]$检测三元组($[\![a]\!]$，$[\![b]\!]$，$[\![c]\!]$)的正确性：

1. 抽取 $s\leftarrow\mathcal{F}_{\text{Rand}}(\mathbb{F})$。

2. 调用$\Pi_{[\![\cdot]\!]}$. LinComb 计算$[\![\rho]\!]=s\cdot[\![a]\!]-[\![\hat{a}]\!]$。

3. 调用$\Pi_{[\![\cdot]\!]}$. Open 打开$[\![\rho]\!]$得到 ρ。

4. 调用$\Pi_{[\![\cdot]\!]}$. LinComb 计算$[\![\sigma]\!]=s\cdot[\![c]\!]-[\![\hat{c}]\!]-[\![b]\!]\cdot\rho$。

5. 运行$\Pi_{[\![\cdot]\!]}$. Check($[\![\rho]\!]$,$[\![\sigma]\!]$,ρ,0)，如果$\Pi_{[\![\cdot]\!]}$中止则其中止。

对于P_{old}中原参与方想要退出计算的情形，不需要从三元组生成的开始，即 Multiply 就进行相关处理，相反地，我们只需对 Authenticate 生成的($[\![a]\!]$,$[\![b]\!]$,$[\![c]\!]$)进行处理即可。

Multiply&Authenticate-nD 具体如下。

输入(Multiply & Authenticate,P_{new})，各方P_i($N\leqslant i\leqslant n$)执行：

1. 将分片$a^{(i)}$，$b^{(i)}$，$c^{(i)}$与 MAC 分片$m(a)^{(i)}$，$m(b)^{(i)}$，$m(c)^{(i)}$发送给P_1。

2. P_1更新其三元组分片

$$a^{(1)}=a^{(1)}+\sum_{i=N+1}^{n}a^{(i)}$$

$$b^{(1)}=b^{(1)}+\sum_{i=N+1}^{n}b^{(i)}$$

$$c^{(1)}=c^{(1)}+\sum_{i=N+1}^{n}c^{(i)}$$

3. P_1更新其三元组 MAC 分片：

$$m(a)^{(1)}=m(a)^{(1)}+\sum_{i=N+1}^{n}m(a)^{(i)}$$

$$m(b)^{(1)}=m(b)^{(1)}+\sum_{i=N+1}^{n}m(b)^{(i)}$$

$$m(c)^{(1)}=m(c)^{(1)}+\sum_{i=N+1}^{n}m(c)^{(i)}$$

4 参与方自适应变体

当在P_{old}上的预处理阶段完成后，在线计算开始。参与方变更的请求在在线阶段随时出现。我们在第 4 节已经给出了面向各种参与方变更的情况的处理办法，其目的是为了将已经生成的预处理数据重新运用于变更后的参与方集合P_{new}，而无需重新运行整个预处理阶段，避免数据与时间资源的浪费。同时，我们对预处理阶段各组件的微调没有损害 MASCOT 原本的安全性，即在 n 个参与方中出现最多 $n-1$ 个恶意参与方时仍能保证诚实参与方的数据隐私。

基于第 4 节对 MASCOT 协议中基本组件的微调，我们给出该协议增加了参与方自适应性的变体版本，如图 2 所示。蓝色部分表示 MASCOT 协议正常运行，橘色部分表明当新参与方请求加入计算时的运行路线，而紫色部分描述当原参与方请求退出计算时的运行路线。

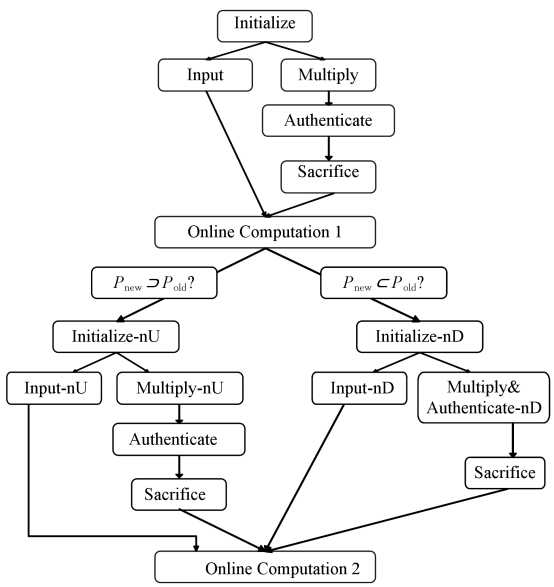

图 2 MASCOT 协议的参与方自适应变体(电子版为彩色)

Fig. 2 Participant-adaptive variant of MASCOT

4.1 安全性分析

MASCOT 协议支持在不诚实的大多数参与方中提供积极安全性。即在 n 个参与方中最多有 $n-1$ 个参与方存在任意攻击行为，例如窃听、篡改等，MASCOT 协议仍然确保不会泄露诚实参与方的信息。为了达到这样的安全特质，MASCOT 以及其他基于 OT 协议的 MPC 协议主要采用了两种技术：信息理论标签(即 MAC 验证)和“牺牲”。我们已经在第 3 节中详细描述了信息理论标签和牺牲技术，这里不再赘述。在 MASCOT 的安全性证明中，主要思想是围绕被攻破参与方提交的任意不合法数据都能被这两种技术识别，进而终止协议。

在本节提出的 MASCOT 协议的参与方自适应变体中，技术细节的设计严格按照原协议的安全准则。即在增加对参与方变更的支持后，变体中对所有的秘密共享数据构建信息理论标签并进行验证，同时利用“牺牲”技术对所有涉及的相关随机进行正确性检测。因此，原 MASCOT 协议的安全性证明仍然适用于当P_{old}变更为P_{new}的情况，参与方自适应的 MASCOT 变体并没有丢失安全性。

4.2 复杂度分析

利用 MASCOT 协议解决多方计算问题时，如果参与方集合发生变更，非常直观的解决方案是重新运行整个协议。这意味着丢弃之前生成的相关随机性、为私有数据和计算中间值构建的秘密共享方案等。为了避免这种情况的发生，以相对较低的代价调整已经生成的各类数据，使其适应新的参与方集合是更好的解决方式。

为了避免太多不必要的细节影响对参与方自适应 MASCOT 变体的复杂度分析，我们不纠结于下层组件具体的运算复杂度，而是以其调用次数评估协议的复杂福。表 2 列出了参与方未变更时的原 MASCOT 协议，以及参与方发生变更后的两种解决方案：1）采用参与方自适应 MASCOT 变体；2）重运行整个 MASCOT 协议的复杂度分析，其中 1 表示可忽略的代价。从表 2 中可以看出，我们设计的 MASCOT 的参与方自适应变体协议很好地填补了参与方发生变更后 MASCOT 协议重新运行前后的操作差异。

表 2 复杂度比较表

Table 2 Complexity comparison

	MASCOT	MASCOT 变体 $\frac{nU}{nD}$	MASCOT 重运行
Initialize	n 次$\Pi_{COPEe}.Initialize$	N-n 次$\Pi_{COPEe}.Initialize$ 1	N 次$\Pi_{COPEe}.Initialize$
Input	n 次$\Pi_{COPEe}.Extend$	N-n 次$\Pi_{COPEe}.Extend$ 1	N 次$\Pi_{COPEe}.Extend$
Multiply	n^2 次$\Pi_{ROT}^{\kappa,k}$	N^2-n^2 次$\Pi_{ROT}^{\kappa,k}$ 1	N^2 次$\Pi_{ROT}^{\kappa,k}$
Authenticate	$5n$ 次$\Pi_{COPEe}.Extend$	$5N$ 次$\Pi_{COPEe}.Extend$ 1	$5N$ 次$\Pi_{COPEe}.Extend$

结束语 MPC 是一个很有吸引力的研究方向。自 20 世纪 80 年代推出以来，MPC 从纯理论研究发展到至今为各类应用程序提供隐私保护的实践工具。我们主要针对在实际应用中部署和使用 MPC 协议——MASCOT 时出现的实际用户需求提供解决方案。该需求是，协同计算任务中的各方可能会因为各种原因发生变化。而我们只需对 MASCOT 的原组件进行较小幅度的调整，就能提供 MASCOT 的参与方自适应版本以满足这种需求。同时，这种调整并不损害 MASCOT 的计算效率与安全水平。因此，我们提出的变体协议由于其高度灵活性，可以被应用到更为广泛的实际场景中。

参 考 文 献

[1] YAO A. Protocols for Secure Computations (Extended Abstract)[C]// IEEE Annual Symposium on Foundations of Computer Science(FOCS). 1982:160-164.

[2] LIPMAA H, ASOKAN N, NIEMI V. Secure Vickrey Auctions without Threshold Trust[C]// Financial Cryptography. 2002:87-101.

[3] PARKES D, RABIN M, SHIEBER S, et al. Practical Secrecy-Preserving, Verifiably Correct and Trustworthy Auctions[J]. Electronic Commerce Research and Applications, 2006, 7(3):70-81.

[4] RABIN M, MANSOUR Y, MUTHUKRISHNAN S, et al. Strictly-Black-Box Zero-Knowledge and Efficient Validation of Financial Transactions[C]// International Colloquium on Automata, Languages, and Programming (ICALP). 2012:738-749.

[5] KUSTERS R, TRUDERUNG T, VOGT A. A Game-based Definition of Coercion Resistance and its Applications[J]. Journal of Computer Security, 2012, 20(6):709-764.

[6] ZHONG H, HUANG L, LUO Y. A Multi-Candidate Electronic Voting Scheme Based on Secure Sum Protocol[J]. Journal of Computer Research and Development, 2006, 43(8):1405-1410.

[7] LUO Y, XU Z, HUANG L. Secure Multi-party Statistical Analysis Problems and Their Applications[J]. Computer Engineering and Application. 2005, 24:145-147.

[8] LIU J, JUUTI M, LU Y, et al. Oblivious Neural Network Predictions via MiniONN Transformations[C]// ACM Conference on Computer and Communications Security. 2017:619-631.

[9] YAGA D, MELL P, ROBY N, et al. Blockchain Technology Overview: NIST Interagency/Internal Report (NISTIR)-8202[R]. 2018.

[10] WANG T, MA W, LUO W. Information Sharingand Secure Multi-party Computing Model Based on Blockchain[J]. Computer Science, 2019, 46(9):162-168.

[11] KOSBA A, MILLER A, SHI E, et al. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts[C]// IEEE Symposium on Security and Privacy. 2016:839-858.

[12] ZYSKIND G, NATHAN O, PENTLAND A. Decentralizing Privacy: Using Blockchain to Protect Personal Data[C]// IEEE Symposium on Security and Privacy Workshops. 2015:180-184.

[13] YAO A. Howto Generate and Exchange Secrets (Extended Abstract)[C]// IEEE Annual Symposium on Foundations of Computer Science (FOCS). 1986:162-167.

[14] GOLDREICH O. The Foundations of Cryptography-Volume 2: Basic Applications[M]. Cambridge University Press, 2004.

[15] GOLDREICH O, MICALI S, WIGDERSON A. How to Play any Mental Game or A Completeness Theorem for Protocols with-Honest Majority[C]// ACM Symposium on Theory of Computing (STOC). 1987:218-229.

[16] BEN-OR M, GOLDWASSER S, WIGDERSON A. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)[C]// ACM Symposium on Theory of Computing(STOC). 1988:1-10.

[17] BEAVER D, MICALI S, ROGAWAY P. The Round Complexity of Secure Protocols (Extended Abstract)[C]// ACM Symposium on Theory of Computing(STOC). 1990:503-513.

[18] KOLESNIKOV V. GateEvaluationSecret SharingandSecureOne-RoundTwo-PartyComputation[C]// The 11th International Conference on the Theory and Application of Cryptology and Infor-

mation Security (ASIACRYPT). 2005:136-155.

[19] KOLESNIKOV V. Secure Two-party Computation and Communication [D]. Canada:University of Toronto,2006.

[20] BENDLIN R,DAMGARD I,ORLANDI C,et al. Semi-homomorphic Encryption and MultipartyComputation[C]// The 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). 2011:169-188.

[21] DAMGARD I,PASTRO V,SMART N,et al. Multiparty Computation from Somewhat Homomorphic Encryption[C]// The 32nd Annual Cryptology Conference (CRYPTO). 2012:643-662.

[22] CRAMER R,DAMGARD I. On the Amortized Complexity of Zero-Knowledge Protocols[C]// The 29th Annual Cryptology Conference (CRYPTO). 2009:177-191.

[23] DAMGARD 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(ESORICS). 2013:1-18.

[24] KELLER M,ORSINI E,SCHOLL P. MASCOT:Faster Malicious Arithmetic Secure Computation with Oblivious Transfer [C]// ACM Conference on Computer and Communications Security. 2016:830-842.

[25] KELLER M,PASTRO V,ROTARU D. Overdrive:Making SPDZ Great Again[C]// The 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). 2018:158-189.

[26] MOHASSELP,RINDAL P. ABY 3:A Mixed Protocol Framework for Machine Learning[C]// ACM Conference on Computer and Communications Security. 2018:35-52.

[27] BUSCHER N,HOLZER A,WEBER A,et al. Compiling Low Depth Circuits for Practical Secure Computation[C]// European Symposium on Research in Computer Security (ESORICS). 2016:80-98.

[28] WANG X,MALOZEMOFF A J,KATZ J. EMP-toolkit:Efficient Multi-Party computation toolkit[EB/OL]. (2016-11-03) [2016-11-03]. https://github.com/emp-toolkit.

[29] Alexandra Institute. FRESCO-A Framework for Efficient Secure Computation[EB/OL]. (2020-01-07) [2020-01-07]. https://github.com/aicis/fresco.

[30] MOODB,GUPTA D,CARTER H,et al. Frigate:A Validated, Extensible,and Efficient Compiler and Interpreter for Secure Computation[C]// IEEE European Symposium on Security and Privacy (EuroS&P). 2016:112-127.

[31] Multiparty.org Development Team. JavaScript implementation of federated functionalities[EB/OL]. (2020-04-07) [2020-04-07]. https://github.com/multiparty/jiff.

[32] SCHOENMAKERS B. MPyC:Secure Multiparty Computation in Python [EB/OL]. (2020-06-06) [2020-06-06]. https://github.com/lschoe/mpyc.

[33] ZAHUR S,EVANS D. Obliv-C:A Language for Extensible Data Oblivious Computation[R]. Cryptology ePrint Archive,Report 2015:1153.

[34] LIU C,WANG X S,NAYAK K,et al. ObliVM:A Programming Framework for Secure Computation[C]// IEEE Symposium on Security and Privacy. 2015:359-376.

[35] ZHANG Y,STEELE A,BLANTON M. PICCO:a General-purpose Compiler for Private Distributed Computation[C]// ACM Conference on Computer and Communications Security. 2013:813-826.

[36] KU Leuven COSIC. SCALE-MAMBA[EB/PL]. (2020-05-06) [2020-05-06]. https://github.com/KULeuven-COSIC/SCALE-MAMBA.

[37] KELLER M. MP-SPDZ:A Versatile Framework for Multi-Party Computation[J]. Cryptology ePrint Archive,Report 2020:521.

[38] BEAVER D. Efficient Multiparty Protocols Using Circuit Randomization[C]// The 11th Annual International Cryptology Conference (CRYPTO). 1991:420-432.

[39] LARRAIA E. Extending Oblivious Transfer Efficiently-or-How to Get Active Security with ConstantCryptographic Overhead [C]// The 3rdInternational Conference on Cryptology and Information Security in Latin America (LATINCRYPT). 2014:368-386.

LI Yan-bin, born in 1988, Ph. D, engineer. Her main research interests include multi-partycomputation, authenticated encryption andsymmetric encryption.

LIU Yu, born in 1981, Ph. D, associate professor, is a member of China Computer Federation. Her main research interests include cryptanalysis and design of symmetric ciphers, quantum cryptanalysis, multi-party computation.