车联网中云链融合隐私保护分布式计算卸载方案

王 强[1)] 赵 全[1),2)] 王 颖[1)] 周福才[1)] 徐 剑[1)]

[1)](东北大学软件学院 沈阳 110819)

[2)](西安交通大学网络空间安全学院 西安 710049)

摘 要 随着车联网技术的迅猛发展,智能汽车中的车载应用越来越多地面临处理时延敏感且计算密集型任务的挑战。尽管车载计算单元具有一定的计算能力,但其有限的资源无法应对时延敏感的复杂任务。针对上述问题,计算卸载至资源丰富的云端是一种可行的解决方案。然而,远程云服务器的传输成本和通信延迟成为卸载时延敏感任务的主要瓶颈。随着移动边缘计算(MEC)兴起,计算能力从中心化的云端转移至网络边缘,降低了延迟,但现有方案仍存在容错性弱、隐私泄露和计算效率低等问题。本文针对这些不足,提出了一种云链融合的隐私保护分布式计算卸载方案(FCOS)。FCOS通过设计冗余分布式计算机制显著提升了容错性,有效减少了单点故障对计算的影响;利用区块链中的智能合约确保计算卸载过程的可验证性与公平性,增强了卸载的安全性;在隐私保护方面,结合同态加密和盲化因子技术,实现了在智能汽车端低计算开销下的数据隐私保护。与现有技术相比,FCOS在云计算阶段处理5000次以内多项式时,计算用时平均降低90.4%;在单点故障率为0%~60%时,10 000次以内的多项式计算用时平均降低了92.9%~96.7%。理论分析与实验结果验证了该方案的高效性、安全性和可靠性。

关键词 车联网;安全计算卸载;区块链;隐私保护;分布式计算

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2025.00433

Cloud-Chain Fusion Privacy-Preserving Distributed Computation Offloading Scheme in Internet of Vehicles

WANG Qiang[1)] ZHAO Quan[1,2)] WANG Ying[1)] ZHOU Fu-Cai[1)] XU jian[1)]

[1)](Software College, Northeastern University, Shenyang 110819)

[2)](School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049)

Abstract With the rapid development of Internet of Vehicles (IoV) technology, smart car applications are increasingly challenged by the need to handle latency-sensitive and computation-intensive tasks. Although on-board computing units possess some processing capabilities, their limited resources often result in inefficiencies when dealing with such complex tasks. Offloading computations to resource-rich cloud servers is a viable solution, but the transmission cost and communication delays between the smart vehicle and remote cloud servers have become major bottlenecks for latency-sensitive tasks. Mobile Edge Computing (MEC) has emerged as a promising solution by shifting computing power from centralized clouds to the network edge, closer to the data source, which reduces latency. However, existing solutions still face challenges such as weak fault tolerance, privacy leaks, and low computational efficiency. To address these issues, this paper proposes a cloud-blockchain integrated privacy-preserving distributed

收稿日期:2024-01-30;在线发布日期:2024-10-24。本课题得到国家自然科学基金项目(62202090, 62173101)、辽宁省博士启动基金(2022-BS-077)、中央高校基本科研业务费(N2417006)、辽宁网络安全执法协同创新中心课题(XTCX2024-015)资助.**王 强**,博士,副教授,硕士生导师,中国计算机学会(CCF)高级会员,主要研究领域为云计算安全、安全多方计算、区块链。E-mail: wangqiang1@mail.neu.edu.cn。**赵 全**,博士研究生,主要研究领域为边缘计算、车载网安全。**王 颖**,硕士研究生,主要研究领域为云计算安全、区块链。**周福才**,博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为密码学、网络安全、可信计算等。**徐 剑**,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为密码学和网络安全。

computation offloading scheme (FCOS). FCOS significantly improves fault tolerance by designing a redundant distributed computing mechanism, which effectively reduces the impact of single-point failure on computation; it utilizes smart contracts in the blockchain to ensure the verifiability and fairness of the computation offloading process, which enhances the security of the offloading; and in terms of privacy protection, it combines the homomorphic encryption and the blinding factor technology to realize the data privacy protection under the low computation overhead at the smart car end. Compared with existing technologies, FCOS reduces the computing time by an average of 90.4% when processing polynomials within 5000 times in the cloud computing stage; when the single-point failure rate is 0%-60%, the computing time of polynomials within 10 000 times is reduced by an average of 92.9%-96.7%. Theoretical analysis and experimental results verify the efficiency, security and reliability of the solution.

Keywords Internet of Vehicles; secure computation offloading; blockchain; privacy-preserving; distributed computation

1 引 言

随着车联网技术的蓬勃发展,车载应用中出现了大量时延敏感的计算密集型业务。而智能汽车的车载计算单元的计算能力和存储资源是有限的,在处理这些高复杂度计算密集型业务时,轻则造成时延较大的问题,重则造成系统的宕机。此外,大量的计算还会给车辆的能耗带来较大负担[1]。因此,如何解决高移动性的车载网络中时延敏感的计算密集型业务是车联网进一步发展的关键问题。

针对上述问题,一个可行的方案是对车载应用中的计算密集型业务进行计算卸载,将车辆本地的大规模计算卸载至云服务器以提高计算效率来降低车辆能耗。但对于时延敏感的计算业务,智能汽车与远程云服务器的传输代价和通信延迟是不可接受的。随着移动边缘计算(Mobile Edge Computing,MEC)技术的出现,研究者将车联网与MEC结合,提出了车辆边缘计算(Vehicular Edge Computing,VEC)[2]框架,其架构如图1所示。

VEC的提出弥补了智能汽车计算能力不足以及能源有限的缺点,增强了数据处理的实时性和可靠性,为智能汽车提供高性能、低延迟的服务保障。同时改进边缘云和远程云的计算方式可进一步提高计算卸载的计算效率。如Guan等人[3]提出一种针对多项式计算的外包计算方案,通过Horner法则的计算方式进一步降低多项式计算的复杂度,从而提高计算卸载的计算效率。

虽然VEC框架具有诸多优点,但在现实应用中仍面临诸多挑战。智能汽车的数据或计算一旦上传至远程云便失去了对它们的直接掌控,所执行的计算全部由第三方不完全可信的远程云代为执行,计算卸载存在数据泄露、隐私泄露和被远程云内部攻击等安全问题[4-5]。在计算结果正确性及完整性方面,远程云可能为了节省计算资源,而随意返回一个不经计算的假结果,也可能遭受外部敌手的攻击返回一个错误的计算结果。

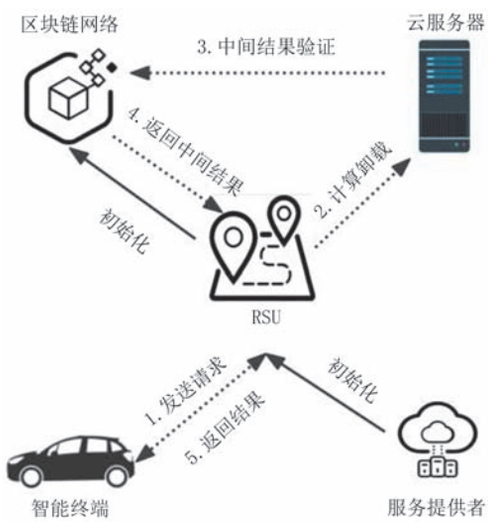

图1 VEC架构图

为实现计算的可追溯性,基于区块链的车联网计算卸载方案被提出[6]。计算将被全部卸载至区块链的智能合约上,利用区块链的不可篡改及可追溯性来保证计算的公开透明。然而以太坊智能合约的执行会产生一定的Gas费用,Gas的消耗与计算量正相关[7]。显然,卸载大规模的计算将为用户造成严重的经济负担[8]。为降低智能合约的Gas消耗,借鉴可验证外包计算[9]思想,云链融合的计算卸载

方案被提出[10]。计算外包至算力强大的远程云，并利用智能合约对远程云计算结果的完整性进行验证[11]。由于可验证外包计算要求用户执行验证的计算开销远小于本地执行计算的开销，因此可有效降低智能合约的Gas消耗。

值得注意的是，区块链的公开透明与隐私保护相矛盾，服务机构的外包计算函数可能包含一些商业机密，并不希望被他人知晓。智能汽车所提交的计算请求可能包含位置信息、行车轨迹和车辆状态等敏感信息也不希望被他人查看。为此，Wang等人[12]提出了一种基于同态加密和区块链技术的车联网隐私保护方案，在不解密的前提下实现密态数据的直接计算。Guo等人[13]利用混淆技术提出了一种基于区块链具有隐私保护的多项式计算卸载方案，保护了多项式的隐私安全。此外，无论在哪种云链融合的计算卸载方案中，远程云往往采用中心化的计算模式，存在单点故障问题[14]。当远程云出现故障时，整个计算任务将陷入阻塞或停滞，严重违背了车载网计算响应低时延的要求。传统的冗余备份、故障恢复和容量规划[15]等方法虽可解决单点故障问题，但难以应用到现有的VEC框架中。其原因为车辆的高移动性和车载网的高可变性等特性使得应用上述方法仍会造成严重的计算时延。因此，亟须一种高效分布式容错计算卸载方案。

针对上述问题，本文提出了云链融合隐私保护分布式计算卸载方案：

（1）设计了一种冗余分布式计算方法：通过在计算卸载中引入冗余机制，不仅提升了计算效率，还有效降低了单点故障带来的影响，增强了系统的鲁棒性和可靠性。

（2）引入智能合约确保计算卸载的可验证性和公平性：利用区块链的智能合约技术，在不增加智能汽车计算开销的情况下，确保计算卸载过程中的可验证性，实现了卸载任务和验证过程的公平性，避免了潜在的恶意节点干扰。

（3）结合同态加密和盲化因子实现隐私保护：在智能汽车端，通过引入同态加密及低开销的盲化因子技术，确保了计算任务和请求的隐私性。该方案在有效保护用户数据隐私的同时，保持了计算和通信的高效性。

（4）理论分析与实验验证：通过理论分析，证明了FCOS在计算效率、系统可靠性及隐私保护性方面具备理论优势。实验结果显示，FCOS与现有技术相比在5000次以内多项式计算中的计算用时平均降低90.4%，在单点故障率为0%～60%时，计算用时平均降低92.8%～96.7%。

2　相关工作

车联网的迅速发展促进了车载应用的繁荣，这些车载应用需要占用智能汽车越来越多的计算资源。由于智能汽车终端计算资源有限，智能车辆难以执行计算密集型任务。为解决上述问题，VEC被引入到车联网中，VEC通过将通信、计算和存储卸载到MEC服务器上，从而实现智能汽车终端计算任务的轻量化。

2.1　可验证计算

由于第三方MEC服务器脱离了数据拥有者的监管，可能因经济原因或外部攻击而返回错误的计算结果。为解决此问题，Gennaro等人[16]于2010年首次提出了可验证计算（Verifiable Computation，VC）的概念。VC允许数据拥有者将密集型计算任务外包至算力强大的云服务器，并以低廉的开销验证计算结果的完整性。由于其具有广阔的应用前景，国内外学者展开了大量研究[5]。按计算类型分类，可分为通用型可验证计算[17]和特定型可验证计算[18]。通用型可验证计算适用于任何函数，一般基于概率可检验证明、零知识证明以及二次扩张程序。

但由于上述方法验证效率远高于本地执行该计算的效率，尚停留在理论研究阶段难以应用。特定型可验证计算通常是为某一类特定计算所量身定制的协议，例如，可验证多项式计算协议[19]、可验证矩阵计算协议[20]、可验证集合计算协议[21]。相比通用型可验证计算，其具有实施难度低、效率高、部署易等优势。

2.2　云链融合的计算卸载方案

虽然可验证计算保证了计算结果的完整性问题，但是难以保证用户和云服务器双方利益的公平性：即便云服务器返回了真实的计算结果，用户仍旧可能抵赖从而拒绝向云服务器支付费用；云服务器返回了伪造的计算结果，却坚称返回真实的计算结果而要求用户支付其费用。为此，研究者提出了基于区块链的计算卸载方案[22]，利用区块链技术来实现交易的公平性。但是将密集型计算任务全部卸载至区块链智能合约，不仅会造成区块链网络响应时间长和吞吐量大等问题，而且还会为用户造成严重的经济负担，例如用户需要支付以太坊智能合约产生的Gas费用。针对此问题，Zhang等人[23]借鉴可

验证计算思想，提出了云链融合的计算卸载方案。将计算卸载至算力强大的远程云，利用智能合约验证计算结果的正确性及完整性。同样，针对矩阵、多项式、集合等类型，研究者们给出了大量云链融合的计算卸载方案。

但是现有方案存在如下两个问题：隐私泄露问题和高时延问题。外包计算和计算请求是以明文形式提交的，不完全可信云服务器可能会泄露其隐私。此外，区块链的公开透明也会加剧隐私泄露的风险。由于车辆的高移动性和车载网的高可变性，现有云链融合计算卸载方案的中心化计算模式难以通过传统的解决方案避免VEC框架下的单点故障问题，不能完全满足车载网的低时延要求。因此，迫切需要一种高效的隐私保护分布式容错计算卸载方案。

2.3 面向多项式的计算卸载方案

多项式运算无论是在科学研究还是现实生活中都是无处不在的，它是矩阵、集合等运算的重要基础。目前面向多项式的云链融合计算卸载方案的研究越来越多。Campanelli等人[24]实现了多项式计算服务的零知识付费，但其方案缺乏设计细节，效率也不如预期，难以应用于车载网络中。同年，Dong等人[25]基于博弈论和智能合约，提出了一种高效的多项式可验证外包计算解决方案。然而，该方案假设参与者间彼此互不勾结，这个假设与区块链分布式系统相违背。Huang等人[26]提出了一种基于承诺采样技术的云链融合计算卸载方案，但需要引入可信第三方执行初始化操作，一旦可信第三方被腐化，方案的安全性将不复存在。Guan等人[3]通过局部采样的方式使用智能合约重复计算Horner法则的部分中间结果来验证计算结果的正确性及完整性。由于采用局部采样的方式，验证的准确性与采样数量正相关。由于不可信的云服务器会泄露外包多项式的隐私性、区块链公开透明的特性也会加剧隐私泄露的风险，Guo等人[13]在Guan等人[3]方案的基础上，采用盲化技术混淆多项式系数来保证其隐私性，但未考虑用户查询请求的隐私安全。同样，上述方案中远程云都采用中心化架构，难以满足车载网络中对计算低时延和容错的要求。

3 基础知识

3.1 同态加密

同态加密（Homomorphic Encryption，HE）是一种加密形式，它允许在不解密的前提下，对两个密文直接运算得到对应明文操作后所对应的密文[27]。对于任意有效的函数f和明文m，有以下性质：

$$f\left(\mathrm{Enc}(m)\right)=\mathrm{Enc}\left(f(m)\right)$$

完全同态加密（Full Homomorphic Encryption，FHE）支持加密数据的无限次加法和乘法运算[28]，适用于任意函数计算，但效率低难以应用[29]；部分同态加密仅支持有限种类的运算，如加法或乘法，效率较高。

Paillier同态加密[30]是Paillier提出的一种加法同态公钥密码系统，满足加法同态和数乘同态。其包含如下三个算法：

（1）$\left(\text{Paillier. pk}, \text{Paillier. sk}\right)\leftarrow\text{Paillier. KeyGen}\left(1^{\lambda}\right)$随机选择两个素数$p$和$q$，使得$gcd\left(pq,(p-1)\cdot(q-1)\right)=1$；计算$N=p\cdot q$和$k=lcm(p-1,q-1)$，其中$lcm$表示最小公倍数；随机选择整数$g\in\mathbb{Z}_{n^2}^{*}$，使得$gcd\left(L\left(g^{k}\bmod N^{2}\right),N\right)=1$，其中函数$L(x)=(x-1)/N$；计算$\mu=\left(L\left(g^{k}\bmod N^{2}\right)\right)^{-1}$；令公钥$\text{Paillier. pk}=(N,g)$，私钥$\text{Paillier. sk}=(k,\mu)$。

（2）$c\leftarrow\text{Paillier. Enc}(\text{Paillier. pk},m)$：对于任意的明文消息$m\in\mathbb{Z}_N$，选择随机数$r\in\mathbb{Z}_N^{*}$，计算密文

$$c=g^{m}r^{N}\bmod N^{2}$$

（3）$m\leftarrow\text{Paillier. Dec}(\text{Paillier. sk},c)$：给定密文$c$，计算得到明文

$$m=\frac{L\left(c^{\kappa}\bmod N^{2}\right)}{L\left(g^{\kappa}\bmod N^{2}\right)}\bmod N=L\left(c^{\kappa}\bmod N^{2}\right)\cdot\mu\bmod N$$

Paillier加法同态表述为：

（1）给定消息m_0和m_1对应的密文c_0和c_1，可计算出m_0+m_1对应的密文：

$$\text{Paillier. Enc}(\text{Paillier. pk},m_0+m_1)=c_0\cdot c_1$$

（2）给定常数k和消息m_0对应的密文c_0，可计算出$k\cdot m_0$对应的密文：

$$\text{Paillier. Enc}\left(\text{Paillier. pk},k,m_0\right)=\underbrace{c_0\cdot\cdots\cdot c_0}_{k}=c_0^{k}$$

3.2 拉格朗日插值法

拉格朗日插值法是一种多项式插值方法，该方法可以给出一个恰好穿过二维平面上若干个已知点的多项式函数。

对某个多项式函数，已知有给定的k个取值点：

$$\left(x_0,y_0\right),\left(x_1,y_1\right),\cdots,\left(x_{k-1},y_{k-1}\right)$$

假设任意两个不同的x_j都互不相同，那么应用拉格朗日插值公式所得到的一元$k-1$次多项式为

$$L(x):=\sum_{j=0}^{k-1}y_j\prod_{i=0,i\neq j}^{k-1}\frac{x-x_i}{x_j-x_i}$$

定义拉格朗日插值法为给定含k个互不相同的取值点的集合w，有下式成立：

$$L(x)=\mathrm{LagrangeForm}(w)$$

根据上述公式可得多项式函数系数向量$\boldsymbol{D}=[d_0\,d_1\cdots d_{n-1}]^T$。

4　FCOS模型的定义

4.1　威胁模型

车联网中云链融合的隐私保护分布式计算卸载方案包含5方实体：服务机构、智能汽车、智能路边单元(Road Side Unit，RSU)、云服务器和区块链网络，其架构图如图2所示。

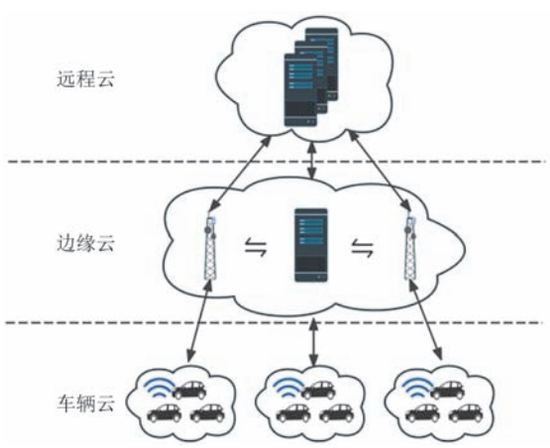

图2　FCOS架构图

服务机构：诚实的计算密集型业务拥有者，将计算密集型业务外包至云服务器，从而为车联网中的智能汽车提供计算服务。服务机构通过提供服务获取利益，错误、不好的服务会影响收益。同时协议执行的过程中服务机构无法控制计算过程进行恶意操作，因此在模型中总是诚实地执行协议。服务机构的部分业务涉及商业机密，存在保护数据隐私的需求。

智能汽车：无恶意地配备传感器、通信模块和车载计算单元等设备的智能汽车，能够发起服务请求并响应车联网中的网络通信。智能汽车在行驶的过程中希望通过发起服务请求获得对应服务。智能汽车的部分数据涉及车辆隐私，不能直接上传。

RSU：分布在通行的道路周围，自身拥有一定的计算能力、存储能力和通信能力，能够通过无线网络与智能汽车和远程云通信的诚实可靠的边缘服务器。RSU具备高级的安全性特征，包括防止恶意攻击、数据篡改和信息泄露等，可采用加密技术来保护通信和数据传输的安全性，确保交通信息和用户隐私不被损害。

云服务器：大型分布式服务器集群，拥有海量的存储能力和计算能力，可为车联网计算密集型业务应用提供计算资源并执行计算卸载。集群是由不完全可信任的计算节点组成的去中心化网络，计算节点的不可信任体现在计算节点可能为了节约资源或恶意攻击随机返回数据作为本节点计算的结果。

区块链网络：由RSU节点组成，主要作用是存储RSU生成的计算模式和验证云服务器计算结果的证据，为网络模型提供稳定、可追溯的信息存储并通过智能合约执行算法。

4.2　形式化定义

本方案主要由8个算法：初始化算法、计算编码算法、请求编码算法、离线初始化算法、分布式计算算法、验证算法、恢复算法和结果解码算法组成，车联网中云链融合隐私保护分布式计算卸载方案可形式化描述为FCOS=(SetUp，EncodeComp，EncodeI，InitOffline，DisComp，Verify，Recover，DecodeR)，各算法形式化定义如下。

(1) 初始化算法：$(pk,sk)\leftarrow\mathrm{SetUp}(F,1^\lambda)$为随机性算法由RSU执行。输入函数簇$F$和安全参数$\lambda$，输出公私钥对$(pk,sk)$。

(2) 计算编码算法：$\sigma_f\leftarrow\mathrm{EncodeComp}(f,pk)$为随机性算法由服务机构执行。输入函数$f\in F$和公钥$pk$，输出编码后的函数$\sigma_f$。

(3) 请求编码算法：$\sigma_x\leftarrow\mathrm{EncodeI}(x,pk)$为随机性算法由智能汽车执行。输入计算输入$x$和公钥$pk$，输出编码输入$\sigma_x$。

(4) 离线初始化算法：$\left(\left\{\sigma_f^{(i)}\right\}_{i\in[n]},\pi\right)\leftarrow\mathrm{InitOffline}(\sigma_f,n)$为随机性算法由RSU执行。输入编码后的函数$\sigma_f$，集群中服务器数量$n$，输出计算模式$\left\{\sigma_f^{(i)}\right\}_{i\in n}$和验证参数$\pi$。

(5) 分布式计算算法：$\sigma_w^{(i)}\leftarrow\mathrm{DisComp}(\sigma_f^{(i)},\sigma_x)$为确定性算法由云服务器执行。输入计算模式$\sigma_f^{(i)}$

和编码输入σ_x，输出分布式计算结果$\sigma_w^{(i)}$。

（6）验证算法：$\{0或1\}\leftarrow \mathrm{Verify}(\sigma_w^{(i)},\Delta_i,\pi)$为确定性算法由区块链网络执行。输入分布式计算结果$\sigma_w^{(i)}$、中间向量$\Delta_i$和验证参数$\pi$。若验证通过，输出1；否则，输出0。

（7）恢复算法：$\sigma_y\leftarrow \mathrm{Recover}(\sigma_w,\sigma_f,\sigma_x,sk)$为确定性算法由RSU执行。输入大小为$s$的分布式计算结果集合$\sigma_w=\{\sigma_w^{(0)},\sigma_w^{(1)},\cdots,\sigma_w^{(s-1)}\}$、编码后的函数$\sigma_f$、编码输入$\sigma_x$和私钥$sk$，输出编码结果$\sigma_y$，其中$s\leqslant n$。

（8）结果解码算法：$y\leftarrow \mathrm{DecodeR}(\sigma_y,pk)$为确定性算法由智能汽车执行。输入编码结果$\sigma_y$和公钥$pk$，输出最终结果$y=f(x)$。

4.3 正确性及安全性定义

一个安全高效的云链融合隐私保护分布式计算卸载方案通常需要满足5个性质：正确性、计算结果的不可伪造性、卸载计算的隐私性、计算请求的隐私性以及恢复阈值的最优性。下面给出上述性质的具体定义。

定义1. 正确性. 模型的正确性简单来说，就是当模型中每个算法都正确执行时，对于每个正确的分布式计算结果$\sigma_w^{(i)}$都能通过验证算法Verify的验证，经恢复算法Recover一定会得到正确的编码结果σ_y，编码结果σ_y经结果解码算法一定会输出正确的计算结果$y=f(x)$。其形式化定义如下：

$$\Pr\left[\begin{array}{l}(pk,sk)\leftarrow \mathrm{SetUp}(F,1^{\lambda});\\ \sigma_f\leftarrow \mathrm{EncodeComp}(f,pk);\\ \sigma_x\leftarrow \mathrm{EncodeI}(x,pk);\\ (\{\sigma_f^{(i)}\}_{i\in[n]},\pi)\leftarrow \mathrm{InitOffline}(\sigma_f,n);\\ \sigma_w^{(i)}\leftarrow \mathrm{DisComp}(\sigma_f^{(i)},\sigma_x):\\ 1\leftarrow \mathrm{Verify}(\sigma_w^{(i)},\Delta_i,\pi)\wedge\\ \sigma_y\leftarrow \mathrm{Recover}(\sigma_w,\sigma_f,\sigma_x,sk)\wedge\\ y\leftarrow \mathrm{DecodeR}(\sigma_y,pk)\end{array}\right]\geqslant 1-negl(\lambda)$$

其中$negl(\lambda)$代表一个可忽略函数。

定义2. 计算结果的不可伪造性. 计算结果的不可伪造性是针对不可信的云服务器而言的。简单地说，要使得云服务器伪造的恶意结果无法通过验证。下面通过安全性实验来定义其安全性，$\mathcal{A}$表示概率多项式时间敌手，$i\in[n]$，构建安全性实验$\mathrm{Exp}_{\mathcal{A}}^{\mathrm{Unforgeability}}[F,\lambda]$如下：

$\mathrm{Exp}_{\mathcal{A}}^{\mathrm{Unforgeability}}[F,\lambda]$:
$(pk,sk)\leftarrow \mathrm{SetUp}(F,1^{\lambda})$;
$\sigma_f\leftarrow \mathrm{EncodeComp}(f,pk)$;
$\left(\{\sigma_f^{(j)}\}_{j\in[n]},\pi\right)\leftarrow \mathrm{InitOffline}(\sigma_f,n)$;
$x\leftarrow \mathcal{A}\left(pk,\sigma_f,\{\sigma_f^{(i)}\}_{i\in[n]},\pi\right)$;
$\sigma_x\leftarrow \mathrm{EncodeI}(x,pk)$;
for $j=1$ **to** s
  $\sigma_w^{(j)}\leftarrow \mathrm{DisComp}(\sigma_f^{(j)},\sigma_x)$;
$\left(i,\sigma_{\hat{w}}^{(i)}\right)\leftarrow \mathcal{A}\left(pk,\sigma_f,\{\sigma_f^{(j)},\sigma_w^{(j)}\}_{j\in[s]},\pi\right)$;
$b\leftarrow \mathrm{Verify}(\sigma_{\hat{w}}^{(i)},\Delta_i,\pi)$;
if $\sigma_{\hat{w}}^{(i)}\neq\sigma_w^{(i)}$ **and** $b=1$
  Output 1;
else
  Output 0;

$\mathrm{Exp}_{\mathcal{A}}^{\mathrm{Privacy}-x}[F,\lambda]$定义了敌手$\mathcal{A}$在已知公钥$pk$、编码函数$\sigma_f$、计算模式$\{\sigma_f^{(i)}\}_{i\in n}$、分布式计算结果$\sigma_w^{(j)}$和验证参数$\pi$的条件下，是否能够伪造一个分布式计算结果$\sigma_{\hat{w}}^{(i)}$，在$\sigma_{\hat{w}}^{(i)}\neq\sigma_w^{(i)}$的情况下通过验证。

对于任意一个概率多项式时间敌手$\mathcal{A}$而言，方案满足计算结果的不可伪造性，则

$$\Pr[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{Unforgeability}}[F,\lambda]=1]\leqslant negl(\lambda)$$

直观地说，如果一个FCOS方案满足隐私性，则恶意的云服务器获取不到关于外包计算函数$f\in F$的任何信息以及用户计算请求的任何信息。假设Π是一个安全高效的云链融合隐私保护分布式计算卸载方案FCOS。下面通过模拟（simulation）的方式来定义方案Π的安全性。方案的功能性$\mathcal{F}$:$f\times\perp\times x\rightarrow\perp\times\perp\times f(x)$，其中$\perp$表示一个空字符串。服务机构、云服务器和智能汽车分别向$\mathcal{F}$输入外包计算函数f，$\perp$和查询请求x。如果方案FCOS满足隐私性，那么存在一个模拟器S能在理想事项中模拟出现实世界中方案FCOS的执行过程，并且无人可区分出这两个世界。下面给出现实世界与理想世界的定义。

（1）现实世界：该世界中包含四个实体：服务机构$\mathcal{O}$、云服务器$\mathcal{C}$、智能汽车$\mathcal{U}$和敌手$\mathcal{A}$。协议执行前，服务机构、云服务器和智能汽车首先接收到各自的输入f、$\perp$和查询请求x。然后，服务机构、云服务器和智能汽车都收到一个随机字符串r和一个辅助的输入z。协议执行后，诚实实体的输出为协议对

应的输出，敌手$\mathcal{A}$的输出为其收集到的全部信息。定义敌手$\mathcal{A}$的输出为$\mathrm{REAL}_{\mathcal{A},\mathcal{C}(z)}^{\Pi}(f,\perp,x)$。

（2）理想世界：该世界包含5方实体：服务机构$\mathcal{O}$、云服务器$\mathcal{C}$、智能汽车$\mathcal{U}$、可信第三方（TTP）和模拟器$\mathcal{S}$。开始时，服务机构$\mathcal{O}$、云服务器$\mathcal{C}$和智能汽车$\mathcal{U}$接收到与现实世界中一样的输入。诚实的实体永远向TTP提交自己的输入，被模拟器$\mathcal{S}$控制的实体可向TTP发送“abort”或任意的输入。TTP返回计算的结果$f(x)$至智能汽车$\mathcal{U}$。如果TTP收到的输入为“abort”，TTP将“abort”发送给用户$\mathcal{U}$。定义模拟器$\mathcal{S}$的输出为$\mathrm{IDEAL}_{\mathcal{S},\mathcal{C}(z)}^{\mathcal{F}}(f,\perp,x)$。

定义3.　隐私性. 如果一个安全高效的云链融合隐私保护分布式计算卸载方案Π具有隐私性，则：

$$\mathrm{IDEAL}_{\mathcal{S},\mathcal{C}(z)}^{\mathcal{F}}(DB,\perp,i)\approx\mathrm{REAL}_{\mathcal{A},\mathcal{C}(z)}^{\Pi}(DB,\perp,i)$$

定义4.　恢复阈值的最优性. RSU能成功恢复出最终的计算结果，使得云服务集群返回有效的分布式计算结果s份最小。

5　方案的详细构建

5.1　初始化算法

初始化算法$\mathrm{SetUp}(F,1^{\lambda})$：以多项式函数为例初始化函数簇$F$并根据安全参数$\lambda$生成公私钥对$(pk,sk)$。具体算法执行步骤如下：

（1）服务机构初始化一元$k-1$次多项式函数簇F，$k\in\mathbb{Z}^{+}$。

（2）RSU调用$\mathrm{Paillier.KeyGen}(1^{\lambda})$生成公私钥对$(\mathrm{Paillier.pk},\mathrm{Paillier.sk})$。

（3）选择一个随机数R，使得$R>\max(F)$。

（4）输出公钥$pk=\{\mathrm{Paillier.pk},R\}$，私钥$sk=\mathrm{Paillier.sk}$。

算法1.　初始化

输入：函数F、安全参数λ

输出：公私钥对(pk,sk)

1. $f\leftarrow\mathrm{InitFun}(F)$
2. $(\mathrm{Paillier.pk},\mathrm{Paillier.sk})=\mathrm{Paillier.KeyGen}(\lambda)$
3. $R=\max(f)$
4. $pk=\{\mathrm{Paillier.pk},R\}$
5. $sk=\mathrm{Paillier.sk}$
6. return (pk,sk)

5.2　计算编码算法

计算编码算法$\mathrm{EncodeComp}(f,pk)$：算法的主要工作是对函数$f\in F$进行编码，用来保护函数$f$的数据隐私安全。具体算法执行步骤如下：

（1）不失一般性记函数f的表达式为

$$f(x)=a_0+a_1x^1+\cdots+a_{k-1}x^{k-1}\bmod R\quad(1)$$

其中$a_i\in\mathbb{Z}$，$i\in[k]$。令$A=[a_0\,a_1\cdots a_{k-1}]$为多项式$f$的系数向量。

（2）使用$\mathrm{Paillier.Enc}(\mathrm{Paillier.pk},A)$对系数向量$A$进行编码，得到编码后的结果：

$$\sigma_A=[c_0\,c_1\cdots c_{k-1}]$$

（3）输出编码后的函数：

$$\sigma_f(x)=c_0+c_1x^1+\cdots+c_{k-1}x^{k-1}\quad(2)$$

算法2.　计算编码

输入：函数f、公钥pk

输出：编码后的函数σ_f

1. for $i=0$ to $A.length-1$
2. $\sigma_A[i]=\mathrm{Paillier.Enc}(A[i])$
3. $\sigma_f\leftarrow\mathrm{InitFun}(f,\sigma_A)$
4. return σ_f

5.3　请求编码算法

请求编码算法$\mathrm{EncodeI}(x,pk)$：算法的主要工作是根据公钥pk中随机数R对智能汽车的输入x进行盲化，$R\in\mathbb{Z}^{+}$，$x\in\mathbb{Z}^{+}$，以保护查询请求的隐私。具体算法执行步骤如下：

（1）将公钥pk解析成$\{\mathrm{Paillier.pk},R\}$。

（2）随机选择盲化因子r，使得$r\geqslant R$。

（3）对输入x进行编码，编码方式如下：

$$\sigma_x=x+r\quad(3)$$

（4）输出编码后的输入σ_x。

5.4　离线初始化算法

离线初始化算法$\mathrm{InitOffline}(\sigma_f,n)$：算法的主要工作是将编码函数分解成$n$个向量乘法的计算模式$\{\sigma_f^{(i)}\}_{i\in[n]}$和用于验证计算结果的证据$\pi$，$n\in\mathbb{Z}^{+}$，$n\geqslant\lceil\sqrt{k}\rceil$，$\lceil\sqrt{k}\rceil$为大于等于$\sqrt{k}$的最小整数。具体算法执行步骤如下：

（1）根据编码后的函数σ_f生成编码系数矩阵

$$\Delta=[c_{i,j}]\quad(4)$$

其中$c_{i,j}=c_{i\times n+j}$，$i\in[n]$，$j\in[n]$。

（2）进而得到编码后的函数σ_f表达式为

$$\sigma_f(x)=x_z\Delta z\quad(5)$$

其中$x_z=[1\,x^n\cdots x^{n(n-1)}]$，$z=[1\,x\cdots x^{n-1}]^T$。这里的$x^i$表示用户输入的占位变量。

(3) 将系数矩阵Δ按行分解,可以得到分解后的表达式:

$$\Delta=\left[\Delta_0\,\Delta_1\cdots\Delta_{n-1}\right]^T \tag{6}$$

其中$\Delta_i=\left[c_{i,0}\,c_{i,1}\cdots c_{i,j}\right]^T$。根据上述表达式令$\tilde{\Delta}_i^{(j)}=\Delta_j^T x_i^{j\alpha}$,其中$x_i^{j\alpha}$表示计算编码结果时云服务器编号的占位变量,计算卸载在线过程中完成赋值,其中$i\in[n]$,$j\in[n]$,$\alpha\in\mathbb{Z}^+$。令$\delta_i=\left\{\Delta_i^{(0)},\Delta_i^{(1)},\cdots,\Delta_i^{(n-1)}\right\}$,根据小节1.5可得:

$$\tilde{\Delta}_i=\prod_{i\in[n]}\delta_i \tag{7}$$

(4) 根据上述初始化,生成向量乘法的中间编码结果的表达式为:

$$\sigma_y^{(i)}=\tilde{\Delta}_i z \tag{8}$$

从而得到分解后的计算模式$\sigma_f^{(i)}=\left\{i,\tilde{\Delta}_i,x_z,z,\sigma_y^{(i)}\right\}$。

(5) 利用伪随机函数G生成长度为m随机验证向量ρ,m为正整数。

(6) 计算用于验证分布式计算结果的验证证据:

$$\tau=z\rho \tag{9}$$

(7) 输出计算模式$\left\{\sigma_f^{(i)}\right\}_{i\in[n]}$和验证参数$\pi=\left\{\rho,\tau\right\}$。

算法3. 离线初始化

输入:编码后的函数σ_f

输出:计算模式$\left\{\sigma_f^{(i)}\right\}_{i\in[n]}$、验证参数$\pi$

```
for i = 0 to σ_A.length − 1
  for j = 0 to n − 1
    Δ[i,j] = σ_A[i × n + j]
x_z ← [1 x^n ⋯ x^{n(n−1)}]
z ← [1 x ⋯ x^{n−1}]^T
for i = 0 to Δ.length − 1
  for j = 0 to Δ[0].length − 1
    Δ'[i][j] ← Δ[i,j]
create mode Δ̃
for i = 0 to Δ'.length − 1
  Δ̃_i += Δ'[i] × x^{iα}
σ_y^{(i)} ← Δ̃_i × z
σ_f^{(i)} = {i, Δ_i, x_z, z, σ_y^{(i)}}
ρ = RandomVector(m)
compute τ = z × ρ
return {σ_f^{(i)}}_{i∈[n]}, π = {ρ, τ}
```

5.5 分布式计算算法

分布式计算算法DisComp$\left(\sigma_f^{(i)},\sigma_x\right)$:算法的主要工作是根据离线初始化算法生成的计算模式$\sigma_f^{(i)}$计算分布式计算结果。具体算法执行步骤如下:

(1) 解析$\sigma_f^{(i)}$得到云服务器编号i、中间向量$\tilde{\Delta}_i$、编码输入σ_x和中间编码结果表达式$\sigma_y^{(i)}$。

(2)将编码请求σ_x代入到表达式z计算得到$\tilde{z}_i$,计算中间编码结果

$$\sigma_{\tilde{y}}^{(i)}=\tilde{\Delta}_i\tilde{z}_i \tag{10}$$

(3) 输出分布式计算结果$\sigma_{\tilde{w}}^{(i)}=\left(i,\sigma_{\tilde{y}}^{(i)}\right)$。

算法4. 分布式计算

输入:计算模式$\left\{\sigma_f^{(i)}\right\}_{i\in[n]}$、编码输入$\sigma_x$

输出:编码结果$\sigma_{\tilde{w}}^{(i)}$

```
z̃_i ← InitMode(z, σ_x)
σ_ỹ^{(i)} = Δ̃_i × z̃_i
σ_w̃^{(i)} ← (i, σ_ỹ^{(i)})
return σ_w̃^{(i)}
```

5.6 验证算法

验证算法Verify$\left(\sigma_{\tilde{w}}^{(i)},\tilde{\Delta}_i,\pi\right)$:算法的主要工作是使用验证参数$\pi$对云服务器返回的分布式计算结果进行验证。具体算法执行步骤如下:

(1) 解析$\sigma_{\tilde{w}}^{(i)}$得到中间编码结果$\sigma_{\tilde{y}}^{(i)}$并解析验证参数为$\pi=\left\{\rho,\tau\right\}$。

(2) 计算如下验证内容:

$$pf_l=\sigma_{\tilde{y}}^{(i)}\rho \tag{11}$$

$$pf_r=\tilde{\Delta}_i\tau \tag{12}$$

(3) 验证如下等式是否成立:

$$pf_l=pf_r \tag{13}$$

(4) 若成立,输出1,否则输出0。

算法5. 验证

输入:编码结果$\sigma_{\tilde{w}}^{(i)}$、中间向量$\tilde{\Delta}_i$、验证参数π

输出:验证结果0或1

```
for i = 0 to ρ.length − 1
  pf_l[i] = σ_ỹ^{(i)} × ρ[i]
for i = 0 to Δ_i.length − 1
  for j = 0 to τ[0].length − 1
    pf_r[i] = Δ_i[i][ ] × τ[ ][j]
if pf_l = pf_r
  t = 1
else
  t = 0
return t
```

算法6. 恢复

输入：编码结果$\sigma_{\tilde{w}}$、加密后的函数σ_f、编码输入σ_x私钥sk

输出：编码结果σ_y

```
for i = 0 to σ_w̃. length − 1
    w̃[i] = Paillier. Dec(σ_w̃^(i))
D = LagrangeForm(w̃)
x̃_z ← InitMode(x_z, σ_x)
for i = 0 to D. length − 1
    σ_y += x̃_z[i] × D[i]
return σ_y
```

5.7 恢复算法

恢复算法Recover$(\sigma_{\tilde{w}}, \sigma_f, \sigma_x, sk)$：算法的主要工作是对分布式计算结果$\sigma_{\tilde{w}}$进行解密，利用解密后的数据计算编码结果。具体算法执行步骤如下：

（1）接收区块链网络返回的分布式计算结果集合$\sigma_{\tilde{w}}=\left\{\sigma_{\tilde{w}}^{(i)} | i\in P, |P|=s\leqslant n\right\}$，$P$是前$s$个通过验证的云服务器编号的集合。

（2）调用Paillier. Dec$(sk, \sigma_{\tilde{w}})$方法对分布式计算结果$\sigma_{\tilde{w}}$进行解码得到：

$$\tilde{w}=\left\{\left(i, \tilde{y}^{(i)}\right) | i\in P, |P|=s\leqslant n\right\}$$

其中$\tilde{y}^{(i)}$为验证通过的分布式计算结果$\sigma_{\tilde{y}}^{(i)}$对应的明文。

（3）利用解码后的分布式计算结果$(\tilde{w})$通过拉格朗日插值法LagrangeForm$(\tilde{w})$获得对应的多项式系数向量：

$$D=\left[d_0\, d_1 \ldots d_{n-1}\right]^T \tag{14}$$

（4）生成向量$\tilde{x}_z$：

$$\tilde{x}_z=\left[1\ \sigma_x^n \ldots \sigma_x^{n(n-1)}\right] \tag{15}$$

（5）将多项式系数向量D与向量$\tilde{x}_z$右乘计算得到编码结果：

$$\sigma_y=\tilde{x}_z \cdot D \tag{16}$$

（6）输出编码结果σ_y。

5.8 结果解码算法

结果解码算法DecodeR(σ_y, pk)：算法的主要工作是使用随机数r对编码结果σ_y进行解码。具体算法执行步骤如下：

对编码结果σ_y进行解码，得到最终结果：

$$y=\sigma_y \bmod r \tag{17}$$

输出最终结果y。

Remark：使用FHE也可以完成对多项式系数向量A和输入x的编码，将此方案称为FCOS-F。基本流程与FCOS一致。在EncodeComp算法和EncodeI算法中，对多项式系数向量A和输入x进行全同态加密。在Recover算法中，使用FHE的私钥直接解密得到对应结果。

5.9 算法示例

图3选取了一个简单、有代表性的示例以便更具体地理解本方案的细节（实际应用和本文实验选

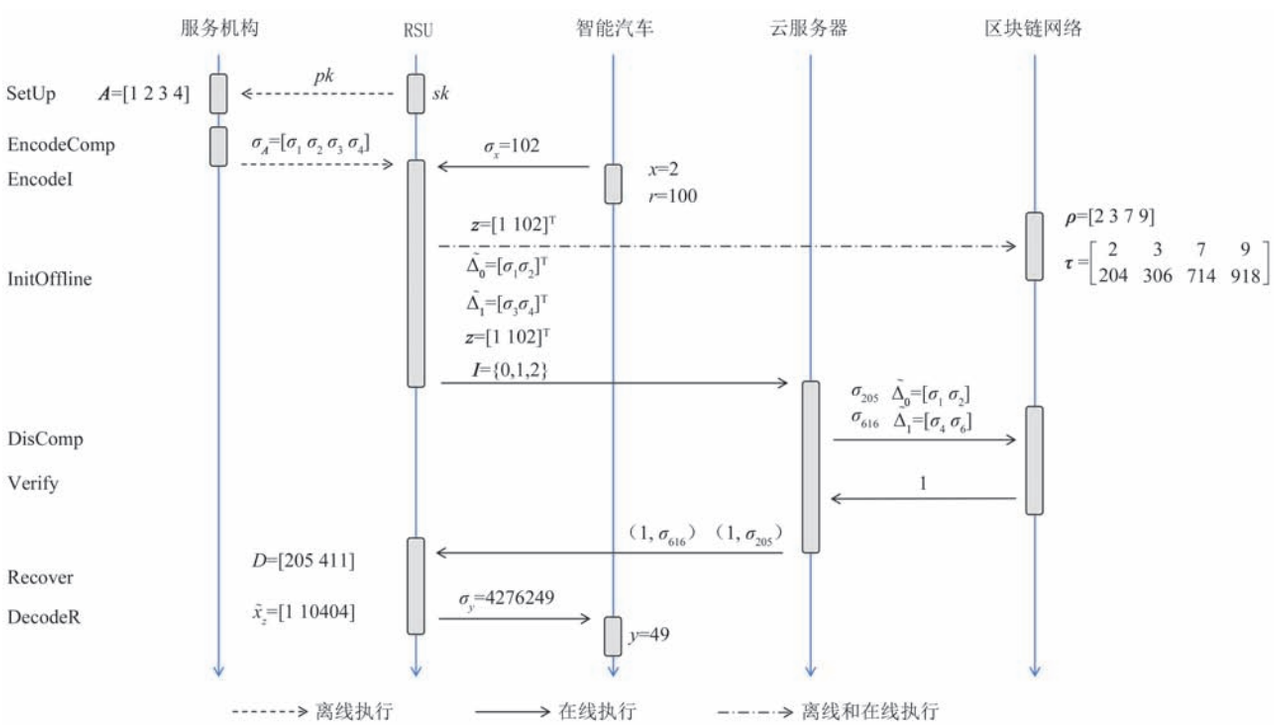

图3　算法流程示例

取数据远大于示例)。示例选取一元三次多项式$f(x)=1+2x+3x^2+4x^3$作为函数f,输入$x=2$。在上述条件下容易求得最终计算结果$f(2)=49$,因此选择盲化因子$r=100$满足方案在此示例的条件。此外为了便于理解,示例中仅以σ_1代表系数1经过Paillier算法加密后的结果。

6 正确性和安全性分析

6.1 正确性

如果方案满足定义1中正确性的定义,则验证算法Verify输出1且恢复算法Recover输出的编码结果σ_y经解码算法能成功得到计算结果$y=f(x)$。因此,需证明等式(13)和(17)成立。下面先给出证明等式(17)成立需要用到的引理1。

引理1. 多项式同余.$f(x)\in\mathbb{Z}[x]$为一元多项式,给定任意随机数$r\geqslant\max(f)$,则下列等式成立

$$f(x+r)\bmod r=f(x)$$

证明. 令函数$g(r)=f(x+r)-f(x)$,故$r=0$是等式$g(r)=0$的根,因此$g(r)=f(x+r)-f(x)$可表示成$g(r)=r\cdot q(r)$,$q(r)$表示关于r的多项式。由于$r\geqslant\max(f)$,显然,$f(x+r)\bmod r=f(x)$成立。

下面证明等式(13)和(17)成立。根据分布式计算算法DisComp中式(10)可得,验证算法Verify中式(11)可写为

$$pf_l=\sigma_{\hat{y}}^{(i)}\rho=\tilde{\Delta}_i\tilde{z}_i\rho$$

同理,由式(7)和(9)可得验证算法Verify中式(12)可以写为

$$pf_r=\tilde{\Delta}_i\tau=\tilde{\Delta}_i\tilde{z}_i\rho$$

显然,根据上述推导等式(13)成立。以下对式(17)的正确性进行分析。

根据Paillier同态加法,恢复算法Verify中式(14)的元素可以写为$d_i=\text{Paillier. Dec}(\Delta_i^Tz)$,进而结合式(15)可以将式(16)写为

$$\begin{aligned}\sigma_y&=\tilde{x}_z\cdot D\\&=\left[1\ \sigma_x^n\cdots\sigma_x^{n(n-1)}\right]\cdot D\\&=\left[1\ \sigma_x^n\cdots\sigma_x^{n(n-1)}\right]\cdot\text{Paillier. Dec}\\&\quad\left(\left[\Delta_0^Tz\ \Delta_1^Tz\cdots\Delta_{n-1}^Tz\right]^T\right)\\&=\sum_{i=0}^{n-1}\sigma_x^{ni}z^T\text{Paillier. Dec}(\Delta_i)\end{aligned}\tag{18}$$

进一步结合式(1)和(5)可得

$$\begin{aligned}\sigma_y&=\sum_{i=0}^{n-1}\sigma_x^{ni}\cdot\left[1\ \sigma_x\cdots\sigma_x^{n-1}\right]\cdot\left[a_{i,0}\ a_{i,1}\cdots a_{i,j}\right]^T\\&=\sum_{i=0}^{n-1}\left[\sigma_x^{ni}\ \sigma_x^{ni+1}\cdots\sigma_x^{ni(n-1)}\right]\cdot\left[a_{i,0}\ a_{i,1}\cdots a_{i,j}\right]^T\\&=\sum_{i=0}^{n-1}a_{i,0}\sigma_x^{ni}+a_{i,1}\sigma_x^{ni+1}+\cdots+a_{i,j}\sigma_x^{ni(n-1)}\\&=\sum_{i=0}^{n-1}a_i\sigma_x^i\end{aligned}$$

其中$i,j\in[n]$,$a_{i,j}=a_{i\times n+j}$。由式(3)可得以下等式成立:

$$\sigma_y=f(\sigma_x)=f(x+r)$$

由引理1可得

$$f(x+r)\bmod r=f(x)=y$$

显然,等式(17)成立。

综上所述,该方案具有正确性。

证毕.

6.2 计算结果的不可伪造性

定理1. 计算结果的不可伪造性. 如果伪随机函数G是安全的,则任意概率多项式时间敌手$\mathcal{A}$不能以不可忽略的概率赢得定义2中安全性实验。

证明. 敌手$\mathcal{A}$返回伪造的编码结果$\sigma_{\hat{w}}^{(i)}\neq\sigma_w^{(i)}$通过验证,其中$\sigma_{\hat{w}}^{(i)}=\{i,\sigma_{\hat{y}}^{(i)}\}$,则有

$$\sigma_{\hat{y}}^{(i)}\rho=\tilde{\Delta}_i\tau\tag{19}$$

并且$\sigma_{\hat{y}}^{(i)}\neq\sigma_y^{(i)}$。

同样,对于通过验证的正确的编码结果有

$$\sigma_y^{(i)}\rho=\tilde{\Delta}_i\tau\tag{20}$$

上述两式可得

$$\left(\sigma_{\hat{y}}^{(i)}-\sigma_y^{(i)}\right)\rho=0\tag{21}$$

由于$\sigma_{\hat{y}}^{(i)}\neq\sigma_y^{(i)}$,故可得向量$\rho$为长度为$m$的0向量,即$\rho=\vec{0}_{1\times m}$。由于向量$\rho$由伪随机函数G产生,并且$\rho\in\mathbb{F}_q^m$,进一步可得

$$\begin{aligned}&\Pr\left(\rho=\vec{0}_{1\times m}\right)\\&=\prod_{p\in\rho,p\in\mathbb{F}_q}\Pr(p=0)\\&=q^{-m}\end{aligned}$$

则有$\Pr\left[\text{Exp}_{\mathcal{A}}^{\text{Unforgeability}}[F,\lambda]=1\right]\leqslant q^{-m}$。

如果敌手$\mathcal{A}$能以概率多项式时间成功伪造出编码结果$\sigma_{\hat{w}}^{(i)}\neq\sigma_w^{(i)}$,则可通过构建一个有效的算法攻破伪随机函数G的安全性。由于向量ρ由伪随机函数G产生,这与伪随机函数G是安全的相矛盾,故该方案在伪随机函数G安全的前提下,满足计算

结果的不可伪造性。

证毕.

6.3　隐私性

定理2.　隐私性. 如果Paillier同态加密具有语义安全，则FCOS方案Π满足定义3所定义的隐私性。

证明.　下面将为被共谋的云服务器$\mathcal{C}$在理想世界构建一个模拟器$\mathcal{S}$，然后再说明理想世界中模拟器$\mathcal{S}$的输出与现实世界中云服务器$\mathcal{C}$的输出是不可区分的。

根据章节5方案的详细构建，现实世界中云服务器$\mathcal{C}$的输出为：

$$\mathrm{REAL}^{\Pi}_{\mathcal{A}_{\mathcal{C}}(z)}(f,\perp,x)=\{\perp,r,pk,\sigma_f(x),\left\{\sigma_f^{(i)}\right\}_{i\in[n]},\pi,\sigma_x,\{0\text{ or }1\}\}$$

其中，0/1表示智能汽车拒绝/接受云服务器$\mathcal{C}$返回的计算结果的编码。

为了模拟现实世界中云服务器$\mathcal{C}$的输出，理想世界中的模拟器$\mathcal{S}$执行如下操作：

(1) 选择一个随机数$\tilde{r}$。

执行调用Paillier. KeyGen(1^{λ})生成公私钥对(Paillier. pk, Paillier. sk)。

(2) 选择一个随机数$\tilde{R}$，使得$\tilde{R}>\max(F)$。

(3) 输出公钥$\widetilde{pk}=\left\{\text{Paillier. pk},\tilde{R}\right\}$，私钥$sk=$Paillier. sk。

(4) 随机选择一个次数为k次的多项式$\tilde{f}(x)=\sum_{i=0}^{k-1}\tilde{a}_i x^i$，其中$\tilde{a}_i\in\mathbb{Z}, i\in[k]$。令$\tilde{\boldsymbol{A}}=[\tilde{a}_0\tilde{a}_1\cdots\tilde{a}_{k-1}]$为多项式$\tilde{f}$的系数向量。

(5) 使用Paillier. Enc(Paillier. pk, $\tilde{A}$)对系数向量$\tilde{A}$进行编码，得到编码后的结果：

$$\tilde{\sigma}_A=[\tilde{c}_0\ \tilde{c}_1\cdots\tilde{c}_{k-1}]$$

(6) 输出编码后的函数：$\sigma_{\tilde{f}}(x)=\tilde{c}_0+\tilde{c}_1x^1+\cdots+\tilde{c}_{k-1}x^{k-1}$。

(7) 将编码后的函数$\sigma_{\tilde{f}}$和n作为输入，执行离线初始化算法InitOffline($\sigma_{\tilde{f}},n$)，其中前4步执行过程一直，随机生成长度为m的验证向量$\tilde{\rho}$，m为正整数。

(8) 计算用于验证分布式计算结果的验证证据$\tilde{\tau}=z\tilde{\rho}$。输出计算模式$\left\{\sigma_{\tilde{f}}^{(i)}\right\}_{i\in[n]}$和验证参数$\tilde{\pi}=\left\{\tilde{\rho},\tilde{\tau}\right\}$。

(9) 从$\mathbb{Z}$中随机选择一个$\tilde{\sigma}_x$作为计算请求。

(10) 如果现实世界中验证算法Verify输出0，则输出0。相反，则输出1。

理想世界中模拟器$\mathcal{S}$的输出为：

$$\mathrm{IDEAL}^{\Pi}_{\mathcal{A}_{\mathcal{C}}(z)}(f,\perp,x)=\{\perp,\tilde{r},\widetilde{pk},\sigma_{\tilde{f}}(x),\left\{\sigma_{\tilde{f}}^{(i)}\right\}_{i\in[n]},\tilde{\pi},\tilde{\sigma}_x,\{0\text{ or }1\}\}$$

(11) 下面证明理想世界中模拟器$\mathcal{S}$的输出与现实世界中敌手$\mathcal{A}$的输出是不可区分的。无论在现实世界还是在理想世界中，输入$\perp$是相同的，因此是不可区分的。随机数r与$\tilde{r}$都是随机选取的，因此也是不可区分的。因为随机数R与$\tilde{R}$都是随机选取的，因此也是不可区分的。现实世界中公钥$pk=\left\{\text{Paillier. pk},R\right\}$，理想世界中公钥$\widetilde{pk}=\left\{\text{Paillier. pk},\tilde{R}\right\}$，因此两世界中的公钥的分布也是不可区分的。又由于理想世界中多项式f的次数与现实世界中随机选择的多项式$\tilde{f}$的次数相等，多项式系数都经过Paillier加密生成编码多项式，由于Paillier加密具有语义安全性，因此现实世界中$\sigma_f(x)$与理想世界中$\sigma_{\tilde{f}}(x)$是不可区分的。同理，现实世界中的计算模式$\left\{\sigma_f^{(i)}\right\}_{i\in[n]}$与理想世界$\left\{\sigma_{\tilde{f}}^{(i)}\right\}_{i\in[n]}$也是不可区分的。由于现实世界中是利用伪随机函数$G$生成长度为$m$随机验证向量$\rho$，理想世界是随机生成长度为$m$随机验证向量$\tilde{\rho}$，故两世界中验证证据是不可区分的。现实世界中查询请求的编码σ_x是由随机数r盲化生成的，盲化因子r私密保存，而理想世界中查询请求的编码是随机选取的，故二者也是不可区分的。对于余下的部分，他们在现实世界与理想世界中是一样的，因此也是不可区分的。综上所述，理想世界中模拟器$\mathcal{S}$的输出与现实世界云服务器$\mathcal{C}$的输出是不可区分的。

证毕.

6.4　恢复阈值的最优性

定理3.　恢复阈值的最优性. 云服务集群返回s份有效的分布式计算结果且能成功恢复出最终的计算结果，则$\min(s)=\left\lceil\sqrt{k}\right\rceil,\left\lceil\sqrt{k}\right\rceil$是大于等于$\sqrt{k}$的最小整数。

证明.　由式(5)和(6)可知Recover算法恢复的关键在于通过s份有效的分布式计算结果求得$b:=\left[\Delta_0^T z\,\Delta_1^T z\ldots\Delta_{s-1}^T z\right]^T$满足式(18)。

分析b的表达式中，只有Δ_i^T是式(4)中系数矩阵的行分量，与恢复阈值相关。由式(2)知多项式系数向量σ_A含k个元素，因此满足式(18)的b中s个

Δ_i^T的元素数量$N=s\times s$需要满足$N\geqslant k$，又$s\in\mathbb{Z}^+$得$s\geqslant\lceil\sqrt{k}\rceil$。

以下证明$s=\lceil\sqrt{k}\rceil$时，Recover算法能够成功恢复结果。由式(8)得s份有效的中间结果表示为

$$\begin{bmatrix}\sigma_y^{(0)}\\ \sigma_y^{(1)}\\ \vdots\\ \sigma_y^{(s-1)}\end{bmatrix}=\begin{bmatrix}0^0 & 0^\alpha & & 0^{\alpha(s-1)}\\ 1^0 & 1^\alpha & & 1^{\alpha(s-1)}\\ \vdots & \vdots & \ddots & \vdots\\ (s-1)^0 & (s-1)^\alpha & & (s-1)^{\alpha(s-1)}\end{bmatrix}\begin{bmatrix}\Delta_0^T z\\ \Delta_1^T z\\ \vdots\\ \Delta_{s-1}^T z\end{bmatrix}$$

想要计算得到b只需求得上式以$\Delta_i^T z$为元素的列向量，其中上式的系数矩阵是范德蒙矩阵，云服务器编号的不同保证了矩阵的行分量是不同的，能够以较大的代价通过方程求解的方式计算得到所求列向量，从而证明了求解该列向量必然有解。而根据上述代数结构不难发现对于第i个中间结果有

$$\sigma_y^{(i)}=\Delta_o^T z+\Delta_1^T z x_i^\alpha+\cdots+\Delta_{s-1}^T z x_i^{\alpha(s-1)}$$

上式中的加法为Paillier同态加法。也就是说$\left(i,\sigma_y^{(i)}\right)$是同态加法下的多项式$h(x)=\sum_{j=0}^{s-1}\Delta_j^T z x^{j\alpha}$上的点，求解$b$只需要求得多项式$h(x)$的系数。已知解决一个只含$s$项的$\alpha(s-1)$次多项式插值问题至少需要$s$个多项式上的点，因此$s$份有效的分布式计算结果可恢复得到式(14)，结合式(18)知Recover算法能够正确恢复最终的计算结果。

综上所述，可证明$\min(s)=\lceil\sqrt{k}\rceil$成立。

证毕.

7 性能分析

7.1 理论分析

本节将本文所提方案FCOS与方案FOPC[3]、POCS[13]、PPOC[31]和FCOS-F在功能方面和性能方面进行对比。在对比前，首先对FOPC、POCS和PPOC进行描述。

FOPC：(1) 外包计算者通过霍纳法则计算多项式并保留霍纳法则计算的中间结果。(2) 验证过程对霍纳法则的计算过程进行抽样，递归完成抽取部分的后续计算。

POCS：(1) 构造一个等比数列对多项式盲化生成两个多项式。(2) 外包计算者通过秦九韶算法计算两个多项式。(3) 区块链上的客户端和挖矿节点对两个多项式进行验证。

PPOC：(1) 使用同态加密算法加密多项式和输入并发送给一个服务器；(2) 服务器使用同态代理重加密和掩码对密文再次加密并发送给另一个服务器，由两个服务器共同完成计算；(3) 使用密文进行公共验证，使用公共验证数据和解密后的数据进行私下验证。

在功能方面，将本方案与各方案间在计算结果的可验证性、计算卸载的形式、数据的隐私保护程度、安全性、容错性和成本进行对比，对比结果如表1所示。本方案相比于其他方案，提出了分布式计算模式提高了方案的容错性；对比FOPC和POCS本方案同时考虑了输入和多项式系数的隐私保护，具有更高的安全性；对比PPOC验证算法基于区块链技术保证了验证过程的不可篡改和公平性并通过对验证算法的设计降低了使用智能合约的成本。

表1 功能对比

方案	可验证	分布式	隐私保护		安全性	容错性	成本
			输入	系数			
FOPC	√	×	×	×	低	低	低
POCS	√	×	×	√	高	低	低
PPOC	√	√	√	√	高	低	—
FCOS-F	√	√	√	√	高	高	高
本方案	√	√	√	√	高	高	低

注：√为支持，×为不支持。

在计算效率方面，将本方案的各算法和其他方案对应算法进行对比，对比结果如表2所示。由于SetUp算法和InitOffline算法可以离线执行不影响方案整体的计算效率，故未列到表中。对于EncodeComp算法POCS通过等比数列的构造对多项式进行盲化，而PPOC和本方案采用加密算法对每个系数加密，因此计算效率的基准不同。PPOC和本方案的EncodeComp算法的复杂度可通过多线程并行加密将由$O(k)^*$降到$O(1)^*$。对于DisComp/Comp算法的复杂度，因本方案通过分布式计算的方式降低了集中计算的复杂度，对比FOPC和POCS算法具有明显优势，而PPOC为防止验证计算中的合谋问题在计算过程中需要重加密，所以计算复杂度更高。对于Verify算法，FOPC的复杂度最低但验证过程具有随机性，而其余方案能够不基于概率的完成对结果的验证。在多项式次数较大时POCS的复杂度低于本方案，而本方案验证矩阵的维度$m<\sqrt{k}$时，PPOC的复杂度更高。本方案的Recover算法的复杂度来自于多项式插值算

法并高于PPOC，但本方案不依赖于特定的插值算法，因此复杂度将随着方案采用的插值算法变化，存在进一步降低的可能。最后对于请求的编码算法EncodeI和解码算法DecodeR，本方案基于盲化而不是加密，因此相比于PPOC和FCOS-F具有明显的优势。

表2　计算效率对比

方案	EncodeComp	EncodeI	DisComp/Comp	Verify	Recover	DecodeR
FOPC	—	—	$O(k)$	$O(\sqrt{nk})$	—	—
POCS	$O(\log_2 k)$	—	$O(2k)$	$O(\log_2 k)$	—	—
PPOC	$O(k)^*$	$O(1)^*$	$O(k)+O(2k)^*$	$O(k)$	$O(\sqrt{k})+O(\sqrt{k})^*$	$O(1)^*$
FCOS-F	$O(k)^*$	$O(1)^*$	$O(\sqrt{k})$	$O(m\sqrt{k})$	$O(k)+O(\sqrt{k})^*$	$O(1)^*$
本方案	$O(k)^*$	$O(1)$	$O(\sqrt{k})$	$O(m\sqrt{k})$	$O(k)+O(\sqrt{k})^*$	$O(1)$

注：k是多项式次数；$0\leqslant n\leqslant 1$；m验证矩阵行维度。*代表基于加密算法轮次的复杂度。

7.2　实验分析

实验中的代码由Python3.6.13和solidity 0.8.17实现，实验环境为Ubuntu 18.04系统，Intel（R）Xeon（R）Platinum 8269CY CPU。除特别声明外，实验中多项式的系数和输入数据均为随机生成的64位整数，安全质数为2048位。实验中的OPCS代表一般多项式计算方案，用于比较Gas消耗并代表中心化计算方案在容错能力方面与本方案进行比较。

7.2.1　与现有方案的对比实验

为了对本方案进行全面的评估，实验对执行了不同程度隐私保护的本方案和无隐私保护的其他方案做了对比，其中FCOS未执行隐私保护算法，FCOS-X只执行了EncodeI算法，FOCS-Y执行了EncodeI算法和EncodeComp算法。另外，由于全同态加密的复杂度过高，严重影响了实验结果的可读性，FCOS-F方案的实验结果未在图中标出。以下将对各实验结果进行分析。

首先对EncodeComp算法的计算开销随安全质数大小和输入数据大小变化的实验结果进行分析。本实验针对EncodeComp算法选取了大小为1024位、2048位和3072位的安全质数和64位、128位和256位的输入进行测试，实验结果如图4所示。

根据实验结果不难发现安全质数的大小对计算开销的影响更大：安全质数大小相同而输入数据大小增大时计算开销的增量较小。对于数据的隐私安全而言，安全质数越大越安全，然而对应的计算开销也会显著增大。因此为了平衡数据安全和计算开销，实验中安全质数选择2048位。

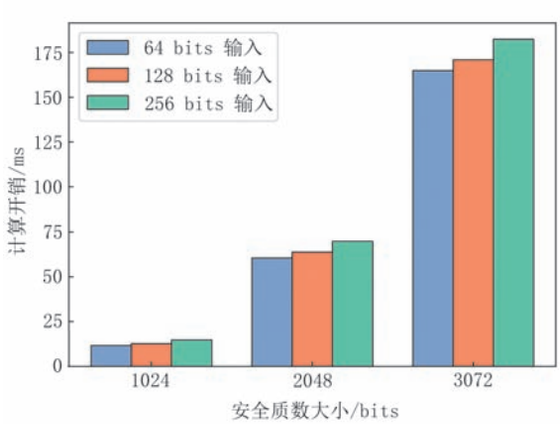

图4　EncodeComp算法计算开销实验结果

与现有方案的对比实验结果如图5所示，以下分析执行隐私保护算法对本方案的影响。由图5(a)、图5(b)和图5(e)不难发现，FCOS-X的图线和FCOS基本重合，说明执行EncodeI算法不影响其他算法的计算开销。而EncodeComp算法对方案的整体影响较大，但在图5(c)中不难发现FCOS-Y在多项式次数较大时计算开销小于PPOC和POCS，这也体现了本方案分布式计算方式的优势。

以下分析与现有方案的对比实验结果。SetUP算法的主要复杂度来自于密钥生成部分，各方案复杂度一致，因此以本方案为例进行简单分析。由图5(a)可得EncodeComp算法的执行对SetUP算法的计算开销影响较大，但随着多项式次数增加而线性增加，整体复杂度相对较低。图5(b)中InitOffline算法不受隐私保护算法执行的影响，计算开销较小。相比于其他方案，虽然本方案需要

额外执行一次InitOffline算法，但在无数据更新的情况下只需要离线执行一次，对方案整体影响较小。由图5(c)可得对于DisComp算法，FCOS和FCOS-X计算用时差距较小，并且远小于所有方案；FCOS-Y与相同隐私保护程度的PPOC相比，5000次内的多项式计算用时平均降低了90.4%。在Recover算法的计算开销上本方案整体高于PPOC，其中主要的计算开销来自插值算法，存在进一步提升的空间。最后DecodeR算法的计算开销较小并不受EncodeComp算法的影响，符合计算卸载方案减少智能汽车计算复杂度和保护隐私的预期目标。

特别的，对于Verify算法的评估度量为Gas消耗(单位为10^{-18}个以太币)、数据大小为256位无符号整数。如图5(d)所示，随着多项式次数变化，情况有所不同。本方案在多项式次数较低时优于FOPC和POCS，在多项式次数较高时优于OPCS。考虑FOPC和POCS验证过程的随机性和OPCS在计算高次多项式时的高消耗，计算卸载方案在验证随机性和Gas消耗方面存在一个平衡。

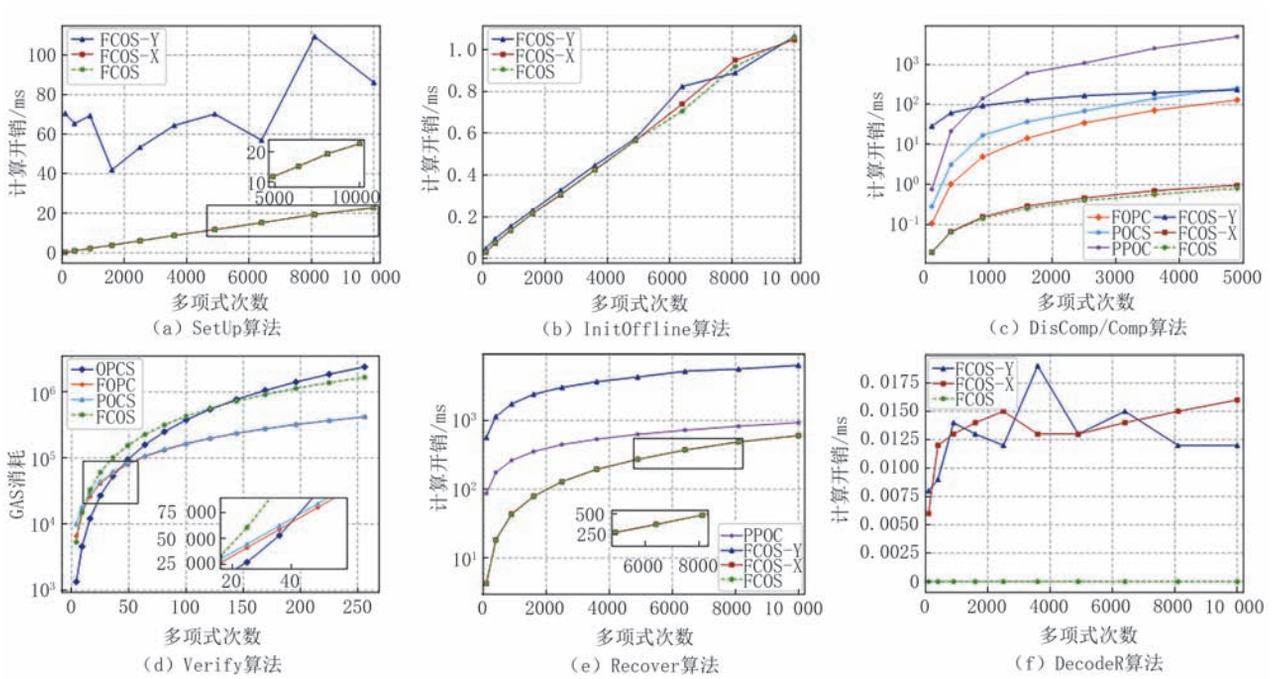

图5 现有方案和本方案的对比实验结果

对于随机性的平衡相对复杂本文不进行更深层次的研究，以下结合本方案和OPCS对Gas消耗的平衡进行量化分析。根据现(2024年4月28日)以太币对美元的汇率关系(1以太币≈3307.94美元)将Gas消耗转为现实货币后的费用部分数据如表3所示。表3列举了4个有代表意义的多项式以美元为单位的计算费用。由表中的信息可知，本方案和OPCS费用相同的多项式在121次和144次之间。因此本方案对计算不低于144次的多项式更有优势，而且随着多项式次数的增加，使用本方案进行计算可以更为显著地降低Gas成本。

表3 计算费用

多项式次数	本方案/美元$\times10^{-9}$	直接计算/美元$\times10^{-9}$	降低费用/美元$\times10^{-9}$
121	1.8669	1.8146	—
144	2.4004	2.5539	0.154
484	14.065	28.180	14.1
576	18.167	39.847	21.7

7.2.2 容错能力

本实验设置4组基于OPCS和本方案的对比实验，分别设置云服务器诚实的返回计算结果的概率为$p=\{100\%,80\%,60\%,40\%\}$，即在实验中每个计算节点会以$p_i$的概率返回正确的计算结果，以$1-p_i$的概率返回一个随机的值作为计算结果代表单点故障发生。分别统计上述实验完成一次计算任务的计算开销作为容错能力的度量标准，容错能力强的方案表现为计算开销受错误计算结果影响小。图6是计算节点以不同的计算正确率返回

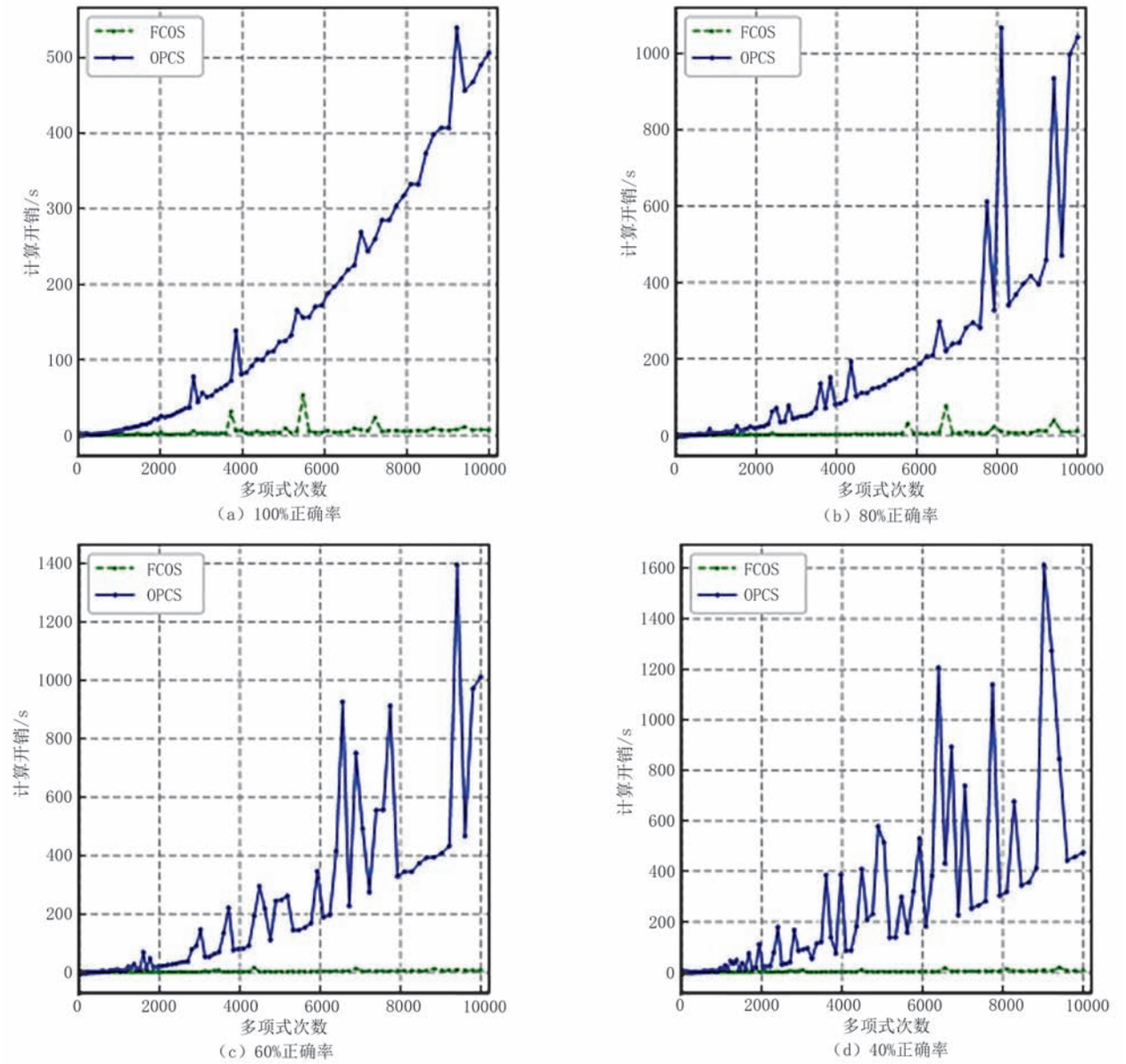

图6　容错能力对比实验结果

计算结果的情况下计算时延随多项式次数增长的变化。不难发现在计算节点不完全可信任的情况下，随着多项式次数的增长，OPCS的计算时延曲线波动越来越大，而本方案的曲线相比于OPCS无明显波动，相对平缓。具体来说，在云服务器单点故障率分别为0%、20%、40%和60%的情况下，本方案相比于OPCS的计算时延平均降低了92.8%、93.7%、96.6%和96.7%。这表明在存在计算错误的前提下，本方案能够以较少的计算时间弥补错误结果造成的损失，即本方案具有较强的容错能力。

8　结　语

本文针对现有车载网中云链融合计算卸载方案所存在的隐私泄露、计算延时大、容错能力弱等问题，提出了一种云链融合的隐私保护分布式计算卸载方案。通过分布式计算来解决单点故障问题并降低计算开销；通过Paillier加密来保护外包函数的隐私性；通过引入盲化因子来保护计算请求的隐私性。理论分析及仿真实验表明本方案的可应用性更好：对车联网场景下所有的多项式计算具有普适性、适用于存在高并发情况和高网络延迟远程云节点的场景。

致 谢 衷心感谢编辑们和评审专家们对本文提出的宝贵意见和建议!

参 考 文 献

[1] Liu Y L, Research on intelligent computing offloading model for the Internet of ssVehicles based on Blockchain [MS. Thesis]. Nanjing University of Posts and Telecommunications, Nanjing, 2022 (in Chinese)
(刘娅利. 基于区块链的车联网智能计算卸载模型研究[硕士学位论文].南京邮电大学,南京,2022)

[2] Li Z Y, Wang Q, Chen Y F, et al. A survey on task offloading research in vehicular edge computing. Chinese Journal of Computers, 2021,44(5):963-982 (in Chinese)
(李智勇,王琦,陈一凡等.车辆边缘计算环境下任务卸载研究综述.计算机学报,2021,44(5):963-982)

[3] Guan Y, Zheng H, Shao J, et al. Fair outsourcing polynomial computation based on the blockchain. IEEE Transactions on Services Computing, 2021, 15(5): 2795-2808

[4] Sun P J. Security and privacy protection in cloud computing: Discussions and challenges. Journal of Network and Computer Applications, 2020, 160: 102642

[5] Yu X, Yan Z, Vasilakos A V. A survey of verifiable computation. Mobile Networks and Applications, 2017, 22: 438-453

[6] Abbes S, Rekhis S. A blockchain-based solution for reputation management in IoV//Proceedings of the 2021 International Wireless Communications and Mobile Computing. Harbin, China, 2021: 1129-1134

[7] Khan M M A, Sarwar H M A, Awais M. Gas consumption analysis of Ethereum blockchain transactions. Concurrency and Computation: Practice and Experience, 2022, 34(4): e6679

[8] Rehman M, Khan Z A, Javed M U, et al. A blockchain based distributed vehicular network architecture for smart cities// Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications. Caserta, Italy, 2020: 320-331

[9] Ozdemir A, Wahby R, Whitehat B, et al. Scaling verifiable computation using efficient set accumulators//Proceedings of the 29th USENIX Security Symposium. Virtual Event, 2020: 2075-2092

[10] Xiao T, Chen C, Pei Q, et al. Consortium blockchain-based computation offloading using mobile edge platoon cloud in internet of vehicles. IEEE Transactions on Intelligent Transportation Systems, 2022, 23(10): 17769-17783

[11] Dorsala M R, Sastry V N, Chapram S. Fair payments for verifiable cloud services using smart contracts. Computers & Security, 2020, 90: 101712

[12] WANG R J, TANG Y C, ZHANG W Q, et al. Privacy protection scheme for internet of vehicles based on homomorphic encryption and block chain technology. Chinese Journal of Network and Information Security, 2020, 6(1): 46-53 (in Chinese)
(王瑞锦,唐榆程,张巍琦等.基于同态加密和区块链技术的车联网隐私保护方案.网络与信息安全学报,2020,6(1):46-53)

[13] Guo Zhen, Zhang Yin, et al. Secure computing outsourcing scheme for polynomial with privacy protection based on blockchain. Journal of Cyber Security, 2021, 6(1): 78-89 (in Chinese)
(郭祯,张银等.基于区块链的具有隐私保护的多项式外包计算方案.信息安全学报,2021,6(1):78-89)

[14] Sohrabi N, Yi X, Tari Z, et al. BACC: Blockchain-based access control for cloud data//Proceedings of the Australasian Computer Science Week Multiconference. Melbourne VIC, Australia, 2020: 1-10

[15] Al-Turkistani H F, AlFaadhel A. Cyber resiliency in the context of cloud computing through cyber risk assessment// Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics. Riyadh, Saudi Arabia, 2021: 73-78

[16] Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers// Proceedings of the Advances in Cryptology-CRYPTO 2010: 30th Annual Cryptology Conference. Santa Barbara, USA, 2010: 465-482

[17] Ahmad H, Wang L, Hong H, et al. Primitives towards verifiable computation: A survey. Frontiers of Computer Science, 2018, 12: 451-478

[18] Gennaro R. Verifiable outsourced computation: A surve// Proceedings of the ACM Symposium on Principles of Distributed Computing. Washington, USA, 2017: 313-313

[19] Sun J, Zhu B, Qin J, et al. Confidentiality-preserving publicly verifiable computation schemes for polynomial evaluation and matrix-vector multiplication. Security and Communication Networks, 2018(1): 5275132

[20] Liu J, Bi J, Li M. Secure outsourcing of large matrix determinant computation. Frontiers of Computer Science, 2020, 14(6): 146807

[21] Li Y, Ghosh D, Gupta P, et al. Prism: private verifiable set computation over multi-owner outsourced databases// Proceedings of the 2021 International Conference on Management of Data. Virtual Event, 2021: 1116-1128

[22] LIU Xue-jiao, SONG Qing-wu, XIA Ying-jie. Secure computation offloading scheme for matrix in Internet of vehicles based on blockchain. Journal of ZheJiang University (Engineering Science), 2023, 57(1): 144-154 (in Chinese)
(刘雪娇,宋庆武,夏莹杰.基于区块链的车联网矩阵计算安全卸载方案.浙江大学学报(工学版),2023,57(1):144-154)

[23] Zhang Y, Deng R H, Liu X, et al. Blockchain based efficient and robust fair payment for outsourcing services in cloud computing. Information Sciences, 2018, 462: 262-277

[24] Campanelli M, Gennaro R, Goldfeder S, et al. Zero-knowledge contingent payments revisited: Attacks and payments for services//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas, USA, 2017: 229-243

[25] Dong C, Wang Y, Aldweesh A, et al. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable

cloud computing//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas, USA, 2017: 211-227

[26] Huang H, Chen X, Wu Q, et al. Bitcoin-based fair payments for outsourcing computations of fog devices. Future Generation Computer Systems, 2018, 78(2): 850-858

[27] Marcolla C, Sucasas V, Manzano M, et al. Survey on fully homomorphic encryption, theory, and applications. Proceedings of the IEEE, 2022, 110(10): 1572-1609

[28] Zhang J, Cheng X, Yang L, et al. Sok: Fully homomorphic encryption accelerators. ACM Computing Surveys, 2024, 56(12): 1-32.

[29] Mishra P K, Rathee D, Duong D H, et al. Fast secure matrix multiplications over ring-based homomorphic encryption. Information Security Journal: A Global Perspective, 2021, 30(4): 219-234

[30] Wei Wenyan, Research on the application of Paillier homomorphic encryption in privacy protection [M.S. Thesis]. Henan Polytechnic University, Jiaozuo, 2017 (in Chinese)
(魏文燕. Paillier同态密码在隐私保护中的应用研究[硕士学位论文].河南理工大学,焦作,2017)

[31] Song B, Zhou D, Wu J, et al. Protecting function privacy and input privacy in the publicly verifiable outsourcing computation of polynomial functions. Future Internet, 2023, 15(4): 152

WANG Qiang, Ph. D., associate professor. His research interests include verifiable computation, secure multi-party computation and blockchain.

ZHAO Quan, Ph. D. candidate. His research interests include edge computing and security in IoV.

WANG Ying, M. S. candidate. Her research interests include security in cloud computing and blockchain.

ZHOU Fu-Cai, Ph. D., professor. His research interests include cryptography, network security, trusted computing, secure cloud storage, software security, basic theory and critical technology in electronic commerce.

XU Jian, Ph. D., professor. His research interests include cryptography and network security.

Background

The rapid development of the Internet of Vehicles has promoted the prosperity of in vehicle applications, which require an increasing amount of computing resources from intelligent vehicles. Due to the limited computing resources of intelligent vehicle terminals, intelligent vehicles are difficult to perform computationally intensive tasks. To solve these problems, VEC has been introduced into the IoV. VEC offloads communication, computing, and storage to MEC servers, thereby achieving lightweight computing tasks for intelligent vehicle terminals. However, due to the detachment of third-party MEC servers from the supervision of data owners, incorrect computing results may be returned due to economic reasons or external attacks. Therefore, many blockchain-based computing offloading solutions have been proposed, which not only achieve public verification but also ensure fairness between users and servers. Existing solutions have two problems: privacy leakage and high latency. Outsourced computing and computing requests are submitted in plain text, and untrusted cloud servers may leak their privacy. In addition, the transparency of blockchain also exacerbates the risk of privacy leakage. Due to the high mobility of vehicles and the high variability of in vehicle networks, the centralized computing mode of existing cloud chain fusion computing offloading solutions is difficult to avoid VEC through traditional solutions. The single point of failure problem under the VEC framework cannot fully meet the low latency requirements of vehicle networks. Therefore, there is an urgent need for an efficient privacy protection distributed fault-tolerant computing offloading scheme.

To solve the existing shortcomings of cloud-chain fusion computation offloading scheme in Internet of Vehicles including high computational delays, weak fault tolerance, and privacy leakage, this paper proposes a cloud-chain fusion privacy-preserving distributed computation offloading scheme (FCOS). Distributed computing is used to solve single point of failure problems and shorten the response time of computing requests; The smart contract is used to ensure the verifiability of the computation offloading process; Paillier encryption is used to protect the privacy of outsourced functions; blinding factor is used to protect the privacy of computational requests, theoretical and simulation experiments have shown that this scheme has better applicability.

This work was supported in part by the National Natural Science Foundation of China under Grant 62202090 and 62173101, by Liaoning Province Natural Science Foundation Medical-Engineering Cross Joint Fund under Grant 2022-YGJC-24, by Doctoral Scientific Research Foundation of Liaoning Province under Grant 2022-BS-077, and by the Fundamental Research Funds for the Central Universities under Grant N2217009.