基于区块链的大规模线性方程组外包计算方案

丁 艳, 王 娜, 杜学绘

信息工程大学 郑州 中国 450001

摘要 随着云计算和大数据技术的发展, 外包计算受到了越来越多的关注。外包计算为资源受限的用户解决复杂的计算问题提供了新的思路, 但用户在外包过程中可能会面临数据隐私泄露、外包计算方案验证效率不高等问题, 且外包计算中存在用户与外包计算者之间公平支付的问题。为解决这些问题, 本文针对科学界和工程领域常见的大规模线性方程组的求解问题, 结合区块链技术, 提出了一种基于区块链的可审计大规模线性方程组求解外包计算方案。首先, 利用稀疏矩阵盲化技术和随机加法分割方法构造了高效的外包计算方案, 实现了用户外包数据的隐私保护和外包结果的可验证; 其次, 基于具有自动化和强制执行特点的智能合约, 设计了一种新的外包计算智能合约, 实现了外包计算结果的公开验证和用户与外包计算者之间的公平支付, 并保障了用户和外包计算者双方的权益; 同时, 借助区块链不可篡改和可溯源等特性, 将外包过程中所涉及的交互信息记录在区块链上, 并提出了一种链上链下协同的可审计外包计算机制, 实现了对外包计算者计算行为的公开审计, 并能对恶意外包计算者进行追溯; 最后, 进行安全性分析和性能分析, 并与现有的方案进行比较, 证明所提方案是安全高效的且能实现用户与外包计算者之间的公平支付。

关键词 线性方程组; 可验证外包计算; 区块链; 可审计; 公平支付

中图法分类号 TP309 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2025.01.07

Large-scale Linear Equations Outsourcing Computing Scheme Based on Blockchain

DING Yan, WANG Na, DU Xuehui

Information Engineering University, Zhengzhou 450001, China

Abstract With the development of cloud computing and big data technology, outsourcing computing has received more and more attention. Outsourcing computing provides a new approach for resource-constrained users to solve complex computational problems. However, users may encounter issues such as data privacy leakage and inefficient verification of outsourcing schemes during the outsourcing process. Additionally, there exists the challenge of fair payment between users and outsourcing computation providers. To solve these problems, and in response to the common problem of solving large-scale linear equation systems in the scientific community and engineering fields, this paper proposes a secure blockchain-based auditable outsourcing computation scheme for solving large-scale linear equation systems in combination with blockchain technology. Firstly, we construct an efficient outsourcing computation scheme using the sparse matrix blinding technique and random addition partitioning method, which realizes the privacy protection of the user's outsourcing data and the verifiable outsourcing results. Secondly, we design a new smart contract for outsourcing computing based on smart contracts with characteristics of automation and enforceability, which realizes the public verification of outsourcing computing results and fair payment between users and outsourcing computation providers, and protects the rights and interests of both users and outsourcing computation providers. At the same time, we take advantage of the blockchain's features such as tampering and traceability to record the interaction information involved in the outsourcing process on the blockchain. Furthermore, we propose an auditable outsourcing computation mechanism that cooperates with the on-chain and off-chain, which realizes the public auditing of the outsourcing computation providers' computation behavior and can trace back the malicious outsourcing computation providers. Finally, we analyze the security and performance of this paper's scheme and compare this paper's scheme with the existing ones to prove that the proposed scheme is secure and efficient and achieves fair payment between users and outsourcing computation computers.

Key words linear systems of equations; verifiable outsourcing computation; blockchain; auditable; fair payment

通讯作者: 王娜, 博士研究生, 教授, Email: twftina_w@126.com。

本课题得到国家自然科学基金(No. 61802436, No. 62102449)、河南省重点研发与推广专项(No. 222102210069)资助。

收稿日期: 2023-05-07; 修改日期: 2023-07-11; 定稿日期: 2024-11-13

1 引言

随着云计算、大数据技术的快速发展, 出现了一种新的计算范式—外包计算, 即拥有有限计算能力的用户将昂贵的计算任务委托给具有强大计算能力的计算者的行为。用户通过按需购买外包计算服务来达到减轻本地计算开销和时间开销的目的。为确保外包计算的可靠性, 外包计算方案至少需要满足结果的可验证性和验证的高效性等基本需求, 以实现可验证的外包计算[1]。

大规模线性方程组的求解问题[2-3], 是科学界和工程领域最基本的代数问题之一。计算机科学、计算流体力学、生命信息科学、地球物理学等领域[4-7]的科学计算问题往往可以归结为一个或一些大规模线性方程组的求解问题。这些线性方程组规模很大、系数密集, 有时甚至会有几百万个未知变量[2]。求解大规模线性方程组对于计算和存储资源有限的用户来说是很难执行的。

目前大规模线性方程组的可验证外包计算方案主要基于冗余策略、密码学方法和矩阵盲化技术, 研究主要集中在计算结果的可验证性、验证的高效性、数据的隐私保护等方面, 未考虑对外包计算者返回错误结果的不诚实计算行为的审计问题和用户与外包计算者之间的公平支付问题。在实际应用中, 公布经常返回错误结果的外包计算者, 对外包计算者的外包计算行为进行审计、溯源, 对于构建安全、可信的外包计算环境是非常有意义的。此外, 在外包支付方面, 目前主要关注保护用户的经济权益, 却忽略了用户在接收到外包计算者的计算结果后, 存在故意抵赖支付的可能。在外包过程中, 在审计外包计算行为的同时能够实现用户与外包计算者之间的公平支付, 是很有意义的一件事。

区块链是近几年最具颠覆性的新型信息技术之一, 其创新性地融合了 P2P 网络、密码学、共识以及智能合约等技术, 具有分布式、开放共享、不可篡改、可溯源、可编程等特点。在外包计算中, 引入区块链, 将区块链作为用户与外包计算者之间交互的平台, 将外包计算任务发布、外包计算结果的验证结果等内容上链, 可实现对外包计算者外包计算行为的审计和全程溯源, 有利于实现对外包计算者的有效监管; 通过设计区块链智能合约完成外包支付, 可实现用户与外包计算者之间的公平支付, 有利于构建可信的外包计算环境。

基于此, 本文提出了一种基于区块链的可审计大规模线性方程组求解外包计算方案, 采用稀疏矩阵盲化与随机分割方法实现大规模线性方程组的安全高效的可验证外包计算, 采用链上链下协同的可审计外包计算机制实现对外包计算行为的全程审计, 设计外包计算智能合约实现验证、支付过程的自动执行和用户与外包计算者之间的公平支付。

本文工作的主要贡献总结如下:

(1) 提出了一种新的线性方程组求解外包计算方案, 利用稀疏矩阵盲化技术[2]与随机分割方法, 实现了外包数据的隐私保护和外包结果的高效验证。

(2) 提出了一种链上链下协同的可审计外包计算机制, 借助区块链[8]公开透明、可溯源的特性, 将审计事务记录在链上, 实现了对外包计算者计算行为的公开审计及对恶意外包计算者的追溯。

(3) 设计了具有外包验证和公平支付功能的外包计算智能合约, 将验证算法写入区块链智能合约, 实现了对外包计算结果的公开验证; 智能合约中预存了用户和外包计算者的资金, 用于支付验证操作和对恶意行为的惩罚, 实现了用户与外包计算者之间的公平支付, 保障了双方的权益。

(4) 安全性分析与性能分析表明, 本文方案在保护用户数据隐私的前提下, 是一种能够实现可验证、审计和公平支付的高效外包计算方案。

2 相关工作

2.1 大规模线性方程组外包计算方案研究

目前大规模线性方程组外包计算方案主要分为冗余策略、密码学方法和盲化技术三种。

基于冗余策略[9-13]的大规模线性方程组外包计算方案的基本思想是用户将同一个经过加密处理的计算任务外包给不同的多个服务器进行冗余计算, 通过对比各个服务器返回的计算结果来验证外包计算结果的正确性。这种方案在服务器不共谋和至少有一台服务器是诚实的假设前提下, 可以保护用户数据隐私性, 降低用户验证部分的计算开销, 但用户的通信负担和支付开销会变大, 且在现实中服务器不共谋的条件往往很难满足。

在密码学领域, 针对外包计算问题的研究大多是从同态、全同态加密的角度出发, 即在密文数据上进行各种计算, 得到的结果在解密后即是原问题的解。Gentry[14]基于同态密码学技术提出求解大规模线性方程组的外包计算协议。Wang 等人[15]提出了一种利用 Paillier 密码系统[16]和迭代方法来求解大规模线性方程组的安全外包机制。同态加密为外包计算及

其结果验证提供了很好的机密性, 但由于同态加密操作需要高内存消耗、高计算成本, 难以在实际外包计算的应用中部署。

盲化技术即通过矩阵变换的方法对用户的隐私数据进行盲化处理。Atallah 等人[17]首次对科学计算和数值计算的安全外包计算问题进行了研究, 提出了针对矩阵乘法和积分计算的数据盲化技术, 以实现对计算数据的隐私保护。胡杏等人[3]提出两次通信的大型线性方程组外包算法, 降低了用户的计算代价, 但用户不能完全验证计算结果的正确性。Chen 等人[2]利用数据盲化和稀疏矩阵变换技术提出了一个安全外包协议, 实现了外包求解线性方程组的可验证性。蔡建兴等人[18]提出了一个基于克罗内克函数和随机置换的可验证大型线性方程组求解的外包计算协议, 该协议降低了用户的空间存储复杂度。Li 等人[19]构造了一种新的有效矩阵加密方案, 在半诚实模型下可实现大规模线性方程组的安全外包存储和计算, 与之前的工作相比, 该方案降低了存储成本。冯达等人[20]构造了一种伪随机的稀疏严格对角优势矩阵生成算法, 并基于该算法提出了一种高效低存储的外包大规模线性方程组方案。

通过上述分析, 可以发现利用矩阵盲化技术所设计的外包计算方案能够保护用户的数据隐私安全, 相比于使用同态加密的外包计算方案计算复杂度明显降低, 且利用该技术构造的针对线性方程组的外包可验证计算方案, 在不需要冗余计算的情况下能够实现对外包计算结果的验证, 故本文利用矩阵盲化技术设计外包计算方案。

2.2 基于区块链的外包计算研究

到目前为止, 基于区块链的外包计算解决方案可以根据其来源分为工业界和学术界两类。

工业界方面, 比较有名的基于区块链的外包计算平台有 Golem[21]、SONM[22]以及 iExec[23]。根据其官网及公开资料显示, 在外包计算结果的验证性方面, Golem 要求客户端需要 8G 以上的内存来验证结果的正确性, 并且不保证外包计算者一定能收到相应的酬劳; SONM 根据信誉评价体系来保证一定的公平性; iExec 则利用贡献度证明(Proof of contribution)、大多数投票(Majority voting)以及信誉积分(Reputation Score)来保证一定的公平性。

在学术界方面, Dong 等人[24]利用博弈论, 提出一种基于智能合约的高效的可验证外包计算解决方案, 但是此方案不能抵抗服务器的共谋攻击。Huang 等人[25]基于比特币和承诺抽样(Commitment-based sampling)技术提出了基于比特币的外包计算解决方案, 但是该方案需要借助可信第三方(Trusted third party, TTP)来解决外包计算双方的不诚实问题。Krol 等人[26]利用以太坊智能合约在可信执行环境(Trusted execution environments, TEE)中实现了对外包计算的安全支付, 但由于可信硬件开发成本较高, 因此该方案的应用场景受限。Hu 等人[27]利用区块链和智能合约构建了一个应用于外包加密数据的具有隐私保护和公平性的搜索方案。由于智能合约的存在, 使得方案中的每一个参与者都能被公平地对待, 并被激励着去做正确的计算。Lin 等人[28]提出了一种许可区块链体系结构来处理双线性配对的安全外包计算问题, 该方案有效解决了使用安全通道和可信服务器的局限性, 但存在计算成本过高的问题。Zhang 等人[29]在以太坊区块链上开发了公平、安全和可验证的外包系统, 有效地执行机器学习中线性回归(Linear regression)算法的训练过程。Guan 等人[30]采用 Horner 方法和区块链技术, 提出了一种新的外包多项式计算方案, 该方案实现了公平支付且是可公开验证, 但并没有对外包计算者的计算行为进行审计。

通过上述分析, 可以发现目前外包计算方案与区块链技术的结合促进了外包计算领域的发展。但大多数应用区块链的外包方案只是将区块链作为可信平台实现对外包结果正确性的验证, 没有针对外包计算者计算行为的审计。且大多数借助区块链实现外包支付功能的方案只考虑了保障用户的权益, 并没有考虑外包计算者的权益。因此, 设计一种既能审计外包计算者的行为又能保障用户和外包计算者双方权益的方案是非常有必要的。

3 预备知识

3.1 稀疏矩阵概念及性质

定义 1. 稀疏矩阵(Sparse matrix)。当矩阵中零元素的个数远大于非零元素的个数, 且零元素的分布没有规律时, 称该矩阵为稀疏矩阵。相反, 当矩阵中大部分元素为非零元素时, 称该矩阵为稠密矩阵。

设 $\boldsymbol{A}$, $\boldsymbol{B}$ 是两个 $n\times n$ 的普通(非稀疏)矩阵, 计算 $\boldsymbol{AB}$ 或 $\boldsymbol{BA}$ 需要进行 n^3 次标量乘法运算[20], 即计算复杂度为 $O(n^3)$。若其中一个矩阵为稀疏矩阵, 不失一般性, 设 $\boldsymbol{A}$ 为每行最多有 ρ 个非零元的 $n\times n$ 稀疏矩阵($\rho << n$), $\boldsymbol{B}$ 为任意 $n\times n$ 矩阵, 则计算 $\boldsymbol{AB}$ 或 $\boldsymbol{BA}$ 至多需要进行 ρn^2 次标量乘法运算。由于 $\rho << n$, 所以此时计算 $\boldsymbol{AB}$ 的复杂度为 $O(n^2)$。在本文中, 用于对用户隐私数据进行加密的稀疏矩阵都为非奇异

矩阵。

3.2 可验证外包计算及其安全需求

3.2.1 可验证外包计算形式化定义

可验证外包计算方案的形式化定义[31-32]如下。

定义 2. 一个可验证外包计算方案由以下五种子算法组成:

(1) $KeyGen(\lambda,F)\to SK$: 基于一个安全参数 λ 和外包计算任务 F, 该算法将生成一个私钥 SK, 此密钥将由外包计算请求者保存并用来加密输入数据。

(2) $ProbGen(SK,x)\to\sigma_x$: 外包计算请求者执行该算法对输入数据进行加密, 该算法将私钥 SK 和外包计算任务的输入 x 作为输入, 返回被编码的加密结果 σ_x 并将其发送给外包计算者。

(3) $Compute(F,\sigma_x)\to\sigma_y$: 外包计算者执行该算法进行外包计算, 该算法将外包计算任务 F 和加密结果 σ_x 作为输入, 产生最终计算结果 y ($y=F(x)$) 的编码输出 σ_y, 并将 σ_y 发送给验证者。

(4) $Verify(SK,\sigma_y)\to True\cup False$: 验证者执行该算法对外包计算结果进行验证, 该算法将私钥 SK 和输出 σ_y 作为输入, 如果 σ_y 是正确的, 则输出 $True$; 否则, 输出 $False$。

(5) $Solve(SK,\sigma_y)\to y$: 外包计算请求者执行该算法对 σ_y 进行解码, 用 SK 生成最终结果 y。

3.2.2 可验证外包计算方案安全需求

可验证外包计算方案需要满足的安全需求包括正确性、隐私性、高效性和可验证性。

(1) 正确性

正确性指外包计算者按照方案诚实运算得到计算结果, 该结果返回给用户一定能够通过验证。

定义 3. (正确性[33])给出验证外包计算者输出结果正确性的实验如下:

其中 A 为敌手。假设该实验中的算法均被正确执行。如果 $Pr[\hat{y}=F(x_i)]\geqslant 1-negli(\lambda)$, 其中 $negli()$ 是关于输入的可忽略函数, 则称外包计算方案的计算结果是正确的。

$$\begin{aligned} SK &\xleftarrow{R} KeyGen(\lambda,F);\\ x_i &\leftarrow \mathcal{A};\\ \sigma_i &\leftarrow ProbGen(SK,x_i);\\ \sigma_x &\leftarrow Compute(F,\sigma_i);\\ \hat{y} &\leftarrow Solve(\sigma_x). \end{aligned}$$

(2) 隐私性

隐私性要求外包计算者不能从用户的输入/输出数据中获取任何敏感信息。

定义 4.(隐私性[33])给出验证外包计算方案隐私性的实验如下:

$$\begin{aligned} SK &\xleftarrow{R} KeyGen(F,\lambda);\\ (x_0,x_1) &\leftarrow \mathcal{A}^{PubProbGen_{SK}(\cdot)};\\ \sigma_0 &\leftarrow ProbGen(SK,x_0);\\ \sigma_1 &\leftarrow ProbGen(SK,x_1);\\ b &\xleftarrow{R} \{0,1\};\\ b' &\leftarrow \mathcal{A}^{PubProbGen_{SK}(\cdot)}(x_0,x_1,\sigma_b). \end{aligned}$$

在上述实验中, $\mathcal{A}$ 为敌手。对于一个安全的外包计算方案, 实验中敌手 $\mathcal{A}$ 的优势定义如下:

$$Adv_{\mathcal{A}}^{UW}(F,\lambda)=\left|Prob[b=b']-\frac{1}{2}\right|。$$

如果对于任意概率多项式时间敌手 $\mathcal{A}$, 满足:

$$Adv_{\mathcal{A}}^{UW}(F,\lambda)\leqslant negli(\lambda),$$

其中 $negli()$ 是关于输入的可忽略函数, 则称外包计算方案具有隐私性。

(3) 高效性

高效性指用户对外包数据进行的预处理和验证者执行的验证操作应该是高效的, 不应有其他复杂的计算操作, 否则也就失去了外包计算的意义。

定义 5. (α-Efficient[34])外包计算方案采用 α-Efficient 的概念来衡量方案的高效性。如果:

① 用户和外包计算者都正确地执行了外包计算协议;

② 对于任意输入, 用户的执行时间都小于或等于执行 F 时间的 α 倍。

则称外包计算方案是 F 的 α-Efficient 高效实现。

(4) 可验证性

可验证性指任何恶意的外包计算者给出的无效输出都无法通过验证, 验证者(可以是用户或其他第三方)都能以不可忽略的概率检测到错误。

定义 6. (β-Checkable[34])外包计算方案采用 β-Checkable 的概念来衡量可验证性。如果:

① 该方案正确实现了计算任务 F 的功能;

② 对任意输入, 外包计算者不遵守外包计算协议的行为都会被以不低于 β 的概率检测出来。

则称该方案是 F 的 β-Checkable 实现。

4 问题定义与需求分析

4.1 问题定义

大规模线性方程组问题的数学表达式如下:

$$\boldsymbol{Ax}=\boldsymbol{b} \tag{1}$$

其中: $\boldsymbol{A}\in \mathrm{R}^{n\times n}$ 是阶为 n 的系数方阵(秩为 n), $\boldsymbol{x}\in\mathbb{R}^n$ 为待求解向量, $\boldsymbol{b}\in\mathbb{R}^n$ 为已知常数向量。通过外包计算求解未知向量 $\boldsymbol{x}$ 是本文所要研究的数学问题。

4.2 需求分析

本文所研究的大规模线性方程组外包计算方案需要满足以下需求:

(1) 外包计算结果的正确性。本文方案需要满足的基本需求之一是定义 3 给出的计算结果正确性, 即对于任意有效的输入 $\boldsymbol{A}$、$\boldsymbol{b}$, 诚实的外包计算者总能输出有效值 $\boldsymbol{x}$, 满足 $\boldsymbol{Ax}=\boldsymbol{b}$。

(2) 用户的数据隐私性。$\boldsymbol{A}$、$\boldsymbol{x}$、$\boldsymbol{b}$ 中可能包含用户的一些私有信息, 需要在确保用户隐私安全的前提下, 外包求解 $\boldsymbol{x}$。本文方案应该满足定义 4 给出的隐私性需求。

(3) 外包计算的高效性。用户对计算任务进行预处理所花费的时间, 连同正确性验证所花费的时间, 要远小于直接计算任务本身所需要的时间, 这样的外包计算才有价值。本文方案应该满足定义 5 给出的高效性需求。

(4) 外包计算结果的可验证性。由于外包计算者不是完全可信的, 故验证者能对外包计算者返回的计算结果的正确性进行验证。本文方案应该满足定义 6 给出的可验证性需求。

(5) 外包计算行为的可审计性。在本文研究的外包计算场景中, 外包计算者是受利益驱动的。外包计算者想以尽可能少的计算成本获得计算酬金, 所以他可能不会认真执行整个计算任务, 随意返回一个结果或故意返回错误的计算结果。因此, 在对外包计算结果的正确性进行验证的同时, 还要对外包计算者的计算行为进行公开审计, 以有助于构建一个安全、可信的外包计算环境。

(6) 公平支付。在实际中, 用户和外包计算者之间的不信任可能导致双方利益冲突。因此, 一个合理的外包计算方案应该保证用户在支付给外包计算者酬金后可以获得其返回的有效的计算结果, 而外包计算者在正确遵循外包计算协议的情况下可以获得奖励, 在不诚实的情况下会受到惩罚。

5 基于区块链的可审计大规模线性方程组外包计算方案

5.1 基于稀疏矩阵盲化与随机分割的大规模线性方程组外包计算方案

对于大规模线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的求解, 本文提出了一种安全高效的可验证外包计算方案。该方案首先利用随机非奇异稀疏矩阵对用户的外包输入数据 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 进行盲化, 再对盲化后的 $\boldsymbol{b}$ 进行随机加法分割, 使得原线性方程组求解问题转换成两个加密线性方程组求解问题, 并交由两个外包计算者分别进行计算。可以证明, 本文所提的外包计算方案在保证了用户输入、输出数据的隐私性的同时实现了高效性与可验证性。

基于稀疏矩阵盲化与随机分割的大规模线性方程组外包计算方案包含一个五元组算法(KeyGen、ProbGen、Compute、Verify、Solve), 如下:

(1) KeyGen: 为了实现大规模线性方程组的外包求解, 用户选择两个随机的非奇异稀疏矩阵 $\boldsymbol{M},\boldsymbol{N}\in\boldsymbol{R}^{n\times n}$。$\boldsymbol{M}$、$\boldsymbol{N}$ 由用户秘密保存, 用来保护已知系数矩阵 $\boldsymbol{A}$ 和常数向量 $\boldsymbol{b}$ 的隐私性。

(2) ProbGen: 用户首先计算 $\boldsymbol{d}=\boldsymbol{Mb}$, 整理式(1)可得 $\boldsymbol{MAx}=\boldsymbol{Mb}$。令 $\boldsymbol{x}=\boldsymbol{Ny}$ (无需计算), 原始的线性方程可以重写为 $\boldsymbol{MANy}=\boldsymbol{Mb}$。然后, 用户计算 $\boldsymbol{H}=\boldsymbol{MAN}$, 完成矩阵盲化。为进一步满足高安全性应用需求, 用户选取一个随机的系数向量 $\boldsymbol{d_1}$, 进行随机加法分割, 将 $\boldsymbol{d}$ 减去 $\boldsymbol{d_1}$ 得 $\boldsymbol{d_2}$, $\boldsymbol{d}=\boldsymbol{d_1}+\boldsymbol{d_2}$。此时, 对求解 $\boldsymbol{Ax}=\boldsymbol{b}$ 的问题经过矩阵盲化和随机加法分割后, 变成求解 $\boldsymbol{Hy}_1=\boldsymbol{d}_1$ 和 $\boldsymbol{Hy}_2=\boldsymbol{d}_2$ 的问题。

(3) Compute: 外包计算者求解 ProbGen 阶段得到的两个函数 $f_1:\boldsymbol{Hy}_1=\boldsymbol{d}_1$ 和 $f_2:\boldsymbol{Hy}_2=\boldsymbol{d}_2$, 得到计算结果 $\boldsymbol{y}_1$、$\boldsymbol{y}_2$。

(4) Verify: 验证者验证 $\boldsymbol{Hy}_1=\boldsymbol{d}_1$ 和 $\boldsymbol{Hy}_2=\boldsymbol{d}_2$ 是否成立并返回验证结果 *True* / *False*。

(5) Solve: 如果验证结果为 *True*, 则用户根据 $\boldsymbol{y}_1$、$\boldsymbol{y}_2$ 的值, 利用私钥 $\boldsymbol{N}$, 计算 $\boldsymbol{x}=\boldsymbol{N}(\boldsymbol{y}_1+\boldsymbol{y}_2)$, 得到初始问题的解。

5.2 链上链下协同的可审计外包计算机制

为了实现对外包计算者外包计算行为的审计, 本文引入区块链作为用户和外包计算者之间交互的平台, 用户和外包计算者之间的所有交互数据都记录在区块链上。但是, 由于采用了块链式存储结构和高冗余的存储机制, 区块链系统的存储容量有限, 一般只有 3～4M[35]。同时, 在实际中大规模线性方程组涉及的数据规模较大。例如, 一个典型的由电磁应用产生的双精度 50000×50000 系统矩阵将产生至少 20GB 的矩阵数据[2]。因此, 大规模的线性方程组矩阵数据不适宜直接存储在区块链上。

基于此, 本文提出了一种链上链下协同的可审计外包计算机制。该机制引入 IPFS(星际文件系统, Inter planetary file system)作为大规模矩阵数据的链

下存储介质, 在链上存储数据的链下存储地址, 通过链上链下协同的方式解决区块链存储容量有限带来的大规模矩阵数据无法直接上链存储的问题。IPFS 是一种基于内容寻址的分布式文件系统, 它以文件的哈希值作为存储地址, 既实现了根据文件内容进行寻址, 又可用于检验文件内容是否被篡改。本文还定义了外包计算审计事务结构, 通过定义要在链上记录的所有交互行为的审计必要信息, 为实现对外包计算行为的审计、溯源提供可信记录支撑。

链上链下协同的可审计外包计算机制(如图 1 所示)由用户、外包计算者、区块链、IPFS 组成。

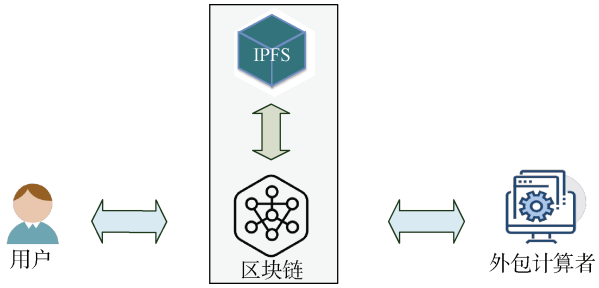

图 1 链上链下协同的可审计外包计算机制架构

Figure 1 Auditable outsourcing computerized architecture with on-chain and off-chain collaboration

(1) 用户(User)。在本文方案中, 用户是大规模线性方程组原始数据的拥有者和外包计算的请求者。用户将加密后的数据上传到 IPFS 存储, 在区块链上部署外包计算智能合约, 并将外包求解 $\boldsymbol{x}$ 的计算任务发布至区块链上, 外包计算智能合约能够实现对外包计算结果正确性的验证以及用户和外包计算者之间的公平支付。用户还将从区块链上获得通过其正确性验证的外包计算结果的存储地址。

(2) 外包计算者(Worker)。外包计算者是本文方案中大规模线性方程组外包求解计算的具体执行者, 其可为任何具有强大算力的计算设备(比如云服务器、个人计算设备等)。外包计算者从区块链上领取外包计算任务, 并在计算完成后, 将计算结果记录上链。本文采用的多外包计算者之间不共谋。

(3) 区块链(Blockchain)。区块链是用户和外包计算者交互的平台, 主要负责部署执行外包计算智能合约以及记录用户与外包计算者之间交互的外包计算审计事务等。此外, 区块链上的挖矿者节点(本文称作验证者节点)会严格执行本文算法对外包计算结果进行正确性验证, 并将验证结果反馈给用户。同时, 验证者节点的验证行为也将上链记录。

(4) IPFS。IPFS 实现用户在外包计算中已知加密数据和外包计算者返回的外包计算结果的链下存储。

为了实现用户与外包计算者之间的交互以及对外包计算行为的审计, 定义以下三类外包计算审计事务:

(1) 外包计算发布事务 $task_{start}$

$$task_{start}=\{uID,f,Haddr,daddr,taskID\}$$

其中, uID 表示用户的唯一标识符, f 表示具体外包计算函数, $Haddr$ 表示密文 $\boldsymbol{H}$ 在 IPFS 中的存储地址, $daddr$ 表示系数向量 $\boldsymbol{d}_1/\boldsymbol{d}_2$ 在 IPFS 中的存储地址, $taskID$ 表示外包计算任务的唯一标识符。

(2) 外包计算结果事务 $task_{end}$

$$task_{end}=\{uID,taskID,wID,yaddr\}$$

其中, wID 表示成功领取外包计算任务的外包计算者的唯一标识符, $yaddr$ 表示外包计算结果 $\boldsymbol{y}_1/\boldsymbol{y}_2$ 在 IPFS 上的存储地址。

(3) 外包结果验证事务 $task_{verify}$

$$task_{verify}=\{uID,taskID,wID,vID,verifyResult\}$$

其中, vID 表示验证者节点的唯一标识符, $verifyResult$ 表示对外包计算结果 $\boldsymbol{y}_1/\boldsymbol{y}_2$ 的验证结果。

5.3 外包计算智能合约

为了实现用户和外包计算者之间的公平支付, 基于具有自动化和强制执行特点的智能合约, 本文设计了具有外包验证和公平支付功能的外包计算智能合约, 该合约具有根据规则自动分配资金的特点, 能根据验证结果实现用户和外包计算者之间的公平支付。

本文将用户部署智能合约所需花费设为 g, 发布外包计算任务需提交的酬金设置为 w, $w=w_1+w_2$, w_1 和 w_2 分别是用户支付给外包计算者和验证者节点的酬金, 根据外包计算任务量制定, 外包计算者领取计算任务需提交的押金设置为 d ($d>w$), 用户/外包计算者只有将 w/d 支付到智能合约才能提交/领取外包计算任务。如果外包计算者返回的计算结果通过正确性验证, 则其将获得用户的酬金, 押金也会被退还; 否则押金将转移给用户作为补偿。

本文设计的外包计算智能合约共包含 7 个函数, 分别是 OT、submitTask、getTask、uploadResult、verify、receiveResult、terminate。这些函数中, 对智能合约调用者的限制通过内置变量 *msg.sender* 来实现, 该变量是从此合约调用方的签名派生出来的。此外, 若某函数被 Payable 修饰, 表示需要用一定金额的资金激活该函数, 如 submitTask 函数和 getTask 函数, 这个金额数值可以通过内置变量 *msg.value* 设置。

(1) 构造函数 OT

函数 OT 是部署智能合约时自动触发执行的构造函数。构造函数中定义了外包计算任务 Task 的结构, 即每一个 Task 包括用户的身份标识 *uID* 、外包计算者的身份标识 *wID* 、外包计算具体函数 f 、密文数据地址 *Haddr*、*daddr* 以及外包计算结果地址 *yaddr* 和验证结果 *verifyResult* ; 定义了 t_u、t_w 两个时间, 分别是用户接收计算结果的时间和外包计算者完成计算的时间; 对外包计算任务集也进行了初始化, 还定义了临时变量 *task _Tmpl* 。

OT 函数如算法 1 所示。

Algorithm 1 OT(g)

```
//构造函数, 部署合约时自动触发执行
struct Task
{
   uID; //用户的身份标识(地址)
   wID; //外包计算者的身份标识
   f; //外包计算具体函数
   Haddr; daddr; //用户数据地址
   yaddr; //外包计算结果地址
   verifyResult; //验证结果
}
t_u; //用户接收计算结果的时间
t_w; //外包计算者计算结果的时间
Task[ ] task; //初始化任务集
Task task _Tmpl; //定义临时变量
```

(2) 任务提交函数 submitTask

submitTask 函数由用户调用, 此函数的输入为 $uID, f, Haddr, daddr$ 。用户调用此函数时需要将酬金 w (包括 w_1和w_2)支付到智能合约中, 这样才能完成一次计算任务的提交, 该酬金 w 将根据智能合约中指定的条件自动转移。用户成功调用此函数后, 其发布的外包计算任务将被写入任务集中, 函数输出计算任务在链上记录的唯一标识 *taskID* 。

submitTask 函数如算法 2 所示。

(3) 任务获取函数 getTask

getTask 函数由外包计算者调用, 此函数的输入为 uID, wID 。外包计算者调用此函数时需要向智能合约支付押金 d (押金 d 也将根据智能合约中指定的条件自动转移), 这样才能获取一个计算任务, 且此外包计算任务未被其他外包计算者领取。外包计算者成功调用此函数时, 合约中的 *msg.sender* 将被设置为此外包计算者 *wID*, 函数输出算法 2 中提到的用户上传的外包计算任务具体信息。

getTask 函数如算法 3 所示。

Algorithm 2 submitTask($uID, f, Haddr, daddr$) Payable

```
//由用户执行, 提交一个任务
require (msg.sender == uID);
require (msg.value == w);
task _Tmpl.uID = uID;
task _Tmpl.f = f;
task _Tmpl.Haddr = Haddr;
task _Tmpl.daddr = daddr;
task _Tmpl.wID =⊥; //⊥ 表示 null
task _Tmpl.yaddr =⊥;
task _Tmpl.verifyResult =⊥;
task.push(task _Tmpl); //增加新的任务
return taskID;
```

Algorithm 3 getTask(uID, wID)Payable

```
//由外包计算者执行, 领取一个任务
require (wID == ⊥); //任务还未被领取
require (msg.value == d); //外包计算者的押金
task _Tmpl.wID = wID;
h = find(uID);
task[h].wID = msg.sender;
return (task[h].f ,task[h].Haddr,
        task[h].daddr);
```

Algorithm 4 uploadResult(*yaddr'*)

```
//由外包计算者执行, 上传结果地址
require (msg.sender==wID and now<t_w);
yaddr = yaddr'; return yaddr;
```

(4) 结果地址上传函数 uploadResult

uploadResult 函数由成功获取外包计算任务的外包计算者调用, 其输入为外包计算结果在 IPFS 中的存储地址。外包计算者需要在规定时间 t_w 内, 将此函数的输出即结果存储地址上传到区块链智能合约中。

uploadResult 函数如算法 4 所示。

(5) 验证函数 verify

verify 函数由区块链中的验证者节点调用, 用

来验证外包计算结果的正确性。此函数的输入为 $f, Haddr, daddr, yaddr$，在完成 $\boldsymbol{Hy}=\boldsymbol{d}$ 的判断后输出 $yaddr, verifyResult$。当验证者节点严格执行合约内容给出验证结果后，酬金 w_2 会自动转移给它。

verify 函数如算法 5 所示。

Algorithm 5 verify($f, Haddr, daddr, yaddr$)

//由区块链验证者节点执行，验证计算结果
require ($msg.sender == verifier$);
h = find(uID);
$\boldsymbol{H} = ipfs.download(Haddr)$;
$\boldsymbol{d} = ipfs.download(daddr)$;
$\boldsymbol{y} = ipfs.download(yaddr)$;
if $\boldsymbol{Hy}=\boldsymbol{d}$ **then** //验证 $\boldsymbol{y}$ 值的正确性
  $verifyResult = True$;
  $task[h].verifyResult = verifyResult$;
  transfer($verifier, w_2$);
else
$verifyResult = False$;
transfer($verifier, w_2$);
return $yaddr, verifyResult$;

(6) 结果接收函数 receiveResult

receiveResult 函数由用户调用，用来接收正确的外包计算结果；其输入为 $yaddr, verifyResult$，当 $verifyResult$ 的值是 $True$ 且用户在时间 t_u 内响应验证结果，用户将接收外包计算结果地址并下载外包计算结果，酬金 w_1 和押金 d 将自动转移给外包计算者；当 $verifyResult$ 的值是 $False$，w_1 和 d 将自动转移给用户。

receiveResult 函数如算法 6 所示。

(7) 任务终止函数 terminate

terminate 函数由用户或外包计算者调用，对未按预期进行的外包计算任务进行资金结算。未按预期进行分为以下几种情况：①没有外包计算者领取用户发布的外包计算任务，则用户调用智能合约里的 terminate 函数，得到酬金 w（w_1 和 w_2）；②外包计算者领取了任务但并未在用户规定的时间 t_w 内提交计算结果，此时用户调用智能合约里的 terminate 函数，获得酬金 w（w_1 和 w_2）和押金 d；③外包计算者在规定时间 t_w 内提交了计算结果，而用户并未在规定时间 t_u 内完成对已进行正确性验证的外包计算结果存储地址的接收，则外包计算者调用 terminate 函数，获得酬金 w_1 和押金 d。

terminate 函数如算法 7 所示。

Algorithm 6 receiveResult($verifyResult, yaddr$)

//由用户执行，接收外包计算结果
require ($msg.sender == uID$);
h = find(uID);
if ($verifyResult == True$ and $t_u < now$) **then**
$task[h].yaddr = yaddr$;
$\boldsymbol{y} = ipfs.download(yaddr)$;
transfer($wID, w_1 + d$);
else
transfer($uID, w_1 + d$);
end if;

Algorithm 7 terminate()

//由用户或外包计算者执行，进行结算
$no\_worker = (msg.sender == uID$
  and $wID == \perp$)//没人领取任务
$submit\_timeout = (msg.sender == uID$
  and $now > t_w$) //外包计算者计算超时
$verify\_timeout = (msg.sender == wID$
  and $now > t_u$)//用户验证超时
if (no_worker or $submit\_timeout$
  or $verify\_timeout$)
transfer($msg.sender$, $this.balance$)
end if;

5.4 工作流程

本文外包计算方案工作流程共分五个阶段，即准备阶段、外包阶段、计算阶段、验收阶段和终止阶段，除准备阶段、计算过程和最终结果的恢复过程在链下完成，方案中其他过程都将在智能合约中完成。本文外包计算方案工作流程图如图 2 所示，下面进行详述。

(1) 第 1 阶段：准备阶段

用户调用 KeyGen 算法和 ProbGen 算法，利用随机选取的非奇异稀疏矩阵 $\boldsymbol{M}$、$\boldsymbol{N}$，将大型线性方程组中包含用户隐私信息的系数矩阵 $\boldsymbol{A}$、常数向量 $\boldsymbol{b}$ 进行盲化，将盲化后的 $\boldsymbol{b}$ 分割为两部分 $\boldsymbol{d}_1$、$\boldsymbol{d}_2$，随后将密文 $\boldsymbol{H}$、$\boldsymbol{d}_1$、$\boldsymbol{d}_2$ 上传至 IPFS 进行存储并得到密文相应的存储地址 $Haddr$、$daddr_1$ 和 $daddr_2$。

(2) 第 2 阶段：外包阶段

用户创建外包计算合约，并将该合约部署到区块链上，区块链返回给用户此合约的标识 $scID$。用户分别调用合约 $scID$ 中的 submitTask 函数，生成两

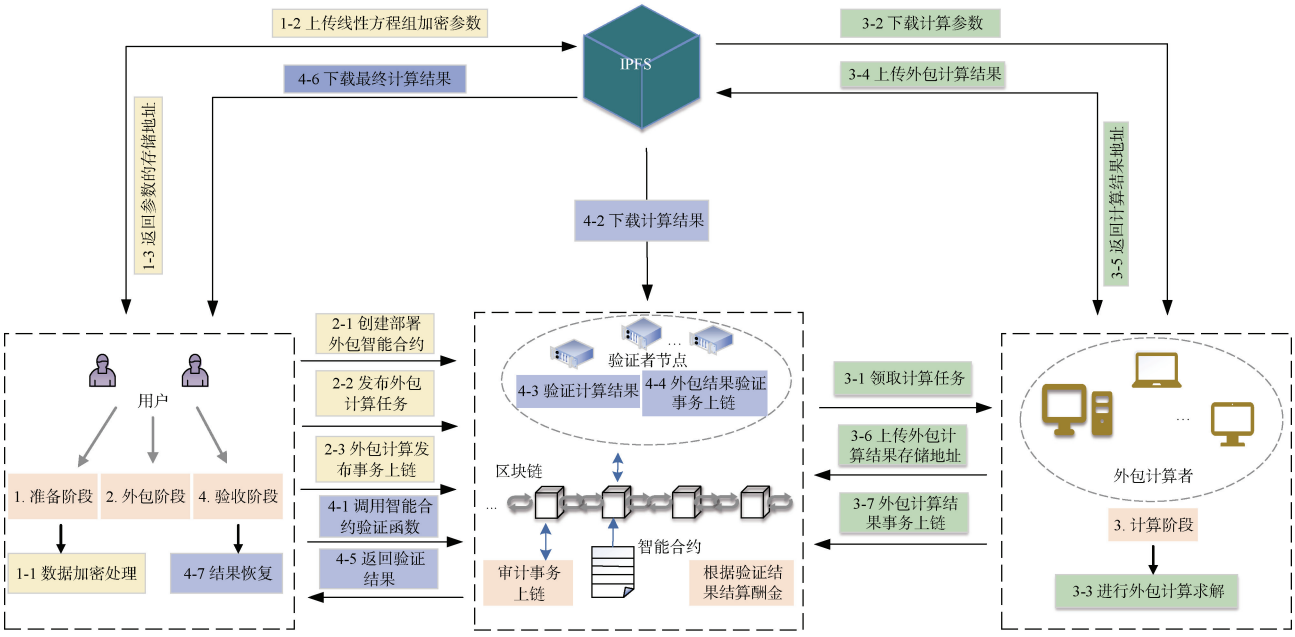

图 2　外包计算方案工作流程图

Figure 2　Outsourcing computation scheme workflow diagram

个外包计算任务, 并分别向合约支付酬金 w。相应的外包计算任务会被提交到区块链上, 用户将得到这两个外包计算任务在链上的标识 $taskID_1$ 和 $taskID_2$。此阶段中用户将生成外包计算发布事务 $task_{start1}$、$task_{start2}$, 并将这两个审计事务记录在区块链上。其中,

$$task_{start1}=\{uID,f,Haddr,daddr_1,taskID_1\},$$

$$task_{start2}=\{uID,f,Haddr,daddr_2,taskID_2\}。$$

(3) 第 3 阶段: 计算阶段

外包计算者首先调用合约 $scID$ 中的 getTask 函数获取外包计算任务($taskID_1$ 或 $taskID_2$)并支付押金 d; 然后, 外包计算者分别根据合约 $scID$ 中密文数据的存储地址获得计算参数 $\boldsymbol{H}$ 和 $\boldsymbol{d}_1/\boldsymbol{d}_2$, 再调用 Compute 算法进行外包计算求值, 并将计算结果上传到 IPFS 进行存储, 获得存储地址 $yaddr_1/yaddr_2$; 最后调用 $scID$ 中的 uploadResult 函数将 $yaddr_1/yaddr_2$ 提交到智能合约, 等待下一步的验证。

在两个外包计算者 wID_1 和 wID_2 分别完成 $taskID_1$ 和 $taskID_2$ 的外包计算任务后, 分别生成外包计算结果事务 $task_{end1}$、$task_{end2}$, 并将这两个审计事务记录在区块链上。其中,

$$task_{end1}=\{uID,taskID_1,wID_1,yaddr_1\},$$

$$task_{end2}=\{uID,taskID_2,wID_2,yaddr_2\}。$$

(4) 第 4 阶段: 验收阶段

区块链中验证者节点调用合约 $scID$ 中的 verify 函数来进行结果正确性验证, 验证者节点分别从 IPFS 中下载 $\boldsymbol{H}$、$\boldsymbol{d}_1/\boldsymbol{d}_2$ 和 $\boldsymbol{y}_1/\boldsymbol{y}_2$, 验证 $\boldsymbol{H}\boldsymbol{y}_1=\boldsymbol{d}_1$、$\boldsymbol{H}\boldsymbol{y}_2=\boldsymbol{d}_2$ 是否成立。由于本文方案设计中的验证者节点是诚实可信的, 故验证节点每返回一次验证结果, 将获得奖励 w_2。此阶段中验证者节点将生成外包结果验证事务 $task_{verify1}$、$task_{verify2}$, 并将这两个审计事务记录在区块链上。其中,

$$task_{verify1}=\{uID,taskID_1,wID_1,vID,verifyResult_1\},$$

$$task_{verify2}=\{uID,taskID_2,wID_2,vID,verifyResult_2\}。$$

然后, 用户将调用 $scID$ 中的 receive 函数, 根据验证结果判断是否接收外包计算结果。若计算结果验证均通过, 用户可根据计算结果的地址 $yaddr_1$、$yaddr_2$ 从 IPFS 中获取 $\boldsymbol{y}_1$、$\boldsymbol{y}_2$ 的值, 调用 Solve 算法, 利用私钥 $\boldsymbol{N}$, 计算 $\boldsymbol{x}=\boldsymbol{N}(\boldsymbol{y}_1+\boldsymbol{y}_2)$ 得到初始问题的解, 同时外包计算者 1 和外包计算者 2 都将获得酬金 w_1 和押金 d; 若某一外包计算结果验证失败, 用户会获得酬金 w_1 和押金 d。

(5) 阶段 5: 终止阶段

如果外包过程没有按照预期进行, 用户或外包计算者将调用 $scID$ 中的 terminate 函数进行资金结算。

6　安全性分析

6.1　可验证外包计算基本安全需求分析

定理 1. 本文外包计算方案的返回结果 $\boldsymbol{x}$ 是正确的。

证明. 假设用户和外包计算者都正确执行方案中的所有算法, 则有:

$$\boldsymbol{d}=\boldsymbol{Mb}$$
$$\boldsymbol{H}=\boldsymbol{MAN}$$
$$\boldsymbol{d}=\boldsymbol{d}_1+\boldsymbol{d}_2$$
$$\boldsymbol{x}=\boldsymbol{N}(\boldsymbol{y}_1+\boldsymbol{y}_2)$$

于是:

$$\begin{aligned}&\boldsymbol{Hy}_1+\boldsymbol{Hy}_2=\boldsymbol{d}_1+\boldsymbol{d}_2\\&\Rightarrow \boldsymbol{MAN}(\boldsymbol{y}_1+\boldsymbol{y}_2)=\boldsymbol{d}\\&\Rightarrow \boldsymbol{MAx}=\boldsymbol{Mb}\\&\Rightarrow \boldsymbol{Ax}=\boldsymbol{b}\end{aligned}$$

所以, $Pr[\boldsymbol{Ax}=\boldsymbol{b}]=1\geqslant 1-negli(\lambda)$。证毕。

定理 2. 本文外包计算方案能够保护外包大规模线性方程组的系数矩阵 $\boldsymbol{A}$、系数向量 $\boldsymbol{b}$ 以及最终结果 $\boldsymbol{x}$ 的隐私安全。

证明. 从第 5 节对外包计算方案的内容阐述可以看到, 本文方案规定的多外包计算者之间是不共谋的。因此, 在整个外包过程中, 外包计算者 1 和外包计算者 2 只能分别获得盲化后的数据 $(\boldsymbol{H},\boldsymbol{d}_1)$ 和 $(\boldsymbol{H},\boldsymbol{d}_2)$。首先证明原始输入向量 $\boldsymbol{b}$ 的隐私安全, 为确保 $\boldsymbol{b}$ 的隐私性, 本文方案采用了随机稀疏矩阵乘法加密和随机加法分割的隐私保护技术。原始的向量 $\boldsymbol{b}$ 首先乘一个随机非奇异稀疏矩阵 $\boldsymbol{M}$: $\boldsymbol{d}=\boldsymbol{Mb}$, 然后为隐藏 $\boldsymbol{d}$, 采用随机加法分割将它随机分解成两个向量 $\boldsymbol{d}_1$ 和 $\boldsymbol{d}_2$: $\boldsymbol{d}=\boldsymbol{d}_1+\boldsymbol{d}_2$, 其中 $\boldsymbol{d}_1$ 是用户随机选取的。由于外包计算者 1 和外包计算者 2 不共谋, 故外包计算者 1 无法恢复向量 $\boldsymbol{d}$, 因为他只有 $\boldsymbol{d}_1$, 对外包计算者 2 可以用相同的方式进行分析。显然对原始向量 $\boldsymbol{b}$ 进行矩阵乘法和矩阵分割等操作是能保证其隐私安全的, 因为经过分割后的 $\boldsymbol{d}_1$、$\boldsymbol{d}_2$ 和 $\boldsymbol{b}$ 是不可区分的。因此, 文本方案能够保护用户输入数据 $\boldsymbol{b}$ 的隐私安全。

下面证明系数矩阵 $\boldsymbol{A}$ 的隐私性。原始线性方程组中的系数矩阵 $\boldsymbol{A}$ 经过加密后变为 $\boldsymbol{H}=\boldsymbol{MAN}$, 设 $\boldsymbol{M}=(m_{ij})$, $\boldsymbol{N}=(n_{ij})$, $\boldsymbol{M}'=(m'_{ij})$, $\boldsymbol{N}'=(n'_{ij})$ 是 4 个由用户生成的随机非奇异稀疏矩阵。给定两个由敌手 A 选择的非奇异密集矩阵 $\boldsymbol{A}=(a_{ij})$、$\boldsymbol{A}'=(a'_{ij})$, 用户计算 $\boldsymbol{H}=\boldsymbol{MAN}=(t_{ij})$ 和 $\boldsymbol{H}'=\boldsymbol{M}'\boldsymbol{A}'\boldsymbol{N}'=(t'_{ij})$, 其中:

$$h_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}m_{ik}\times a_{kl}\times n_{lj}\ ,$$

$$h'_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}m'_{ik}\times a'_{kl}\times n'_{lj}\ 。$$

需要注意的是, 4 个矩阵 $\boldsymbol{M}$、$\boldsymbol{N}$、$\boldsymbol{M}'$ 和 $\boldsymbol{N}'$ 的所有非零元素的数值和位置都是由用户随机选择的, 因此 h_{ij} 和 h'_{ij} 在计算上是无法区分的。因此, 敌手 A 区分 $\boldsymbol{H}$ 和 $\boldsymbol{H}'$ 的优势可以忽略不计。

下面证明最终结果 $\boldsymbol{x}$ 的隐私性。本文设定外包计算者之间不共谋, 外包计算者 1 只能得到线性方程组 $\boldsymbol{Hy}_1=\boldsymbol{d}_1$ 的计算结果 $\boldsymbol{y}_1$, 外包计算者 2 只能得到线性方程组 $\boldsymbol{Hy}_2=\boldsymbol{d}_2$ 的计算结果 $\boldsymbol{y}_2$, 又因为 $\boldsymbol{x}=\boldsymbol{N}(\boldsymbol{y}_1+\boldsymbol{y}_2)$, 故外包计算者 1 和外包计算者 2 不能通过用户发送的数据信息和运算结果获得用户输出数据 $\boldsymbol{x}$ 的敏感信息, 因此, $\boldsymbol{x}$ 会得到很好的隐私保护。

定理 3. 本文外包计算方案是大规模线性方程组的 $O(\frac{1}{n})$ -Efficient 实现。

证明. 在本文提出的外包计算方案中, 用户需要执行两次矩阵向量乘法(这里省略了向量加法运算), 这需要 $O(n^2)$ 计算。此外, 用户还需要计算 $\boldsymbol{H}=\boldsymbol{MAN}$, 这也需要 $O(n^2)$ 计算。而用户直接求解线性方程, 需要 $O(n^3)$ 计算。因此, 本文外包计算方案是大规模线性方程组的 $O(\frac{1}{n})$ -Efficient 实现。

定理 4. 本文外包计算方案是一个 1-Checkable 的可验证方案。

证明. 外包计算者给定最终 $\boldsymbol{y}$ 的值, $\boldsymbol{Hy}$ 的计算复杂度为 $O(n^2)$, 验证者可以验证方程 $\boldsymbol{Hy}=\boldsymbol{d}$ 是否有效。因此, 如果外包计算者在本文方案的计算执行过程中存在不端行为返回错误的 $\boldsymbol{y}$ 值, 则验证者检测出错误 $\boldsymbol{y}$ 值的概率为 1。

用户在当前的外包计算方案中是诚实的, 然而, 在现实生活中他们可能并不诚实。本文使用区块链作为外包计算的审计平台, 且将外包计算结果存储在 IPFS 中, 可有效抑制用户拒绝计算结果这类恶意行为的发生。

此外, 实际应用中会存在方程组无解的可能, 下面考虑外包方程组无解的情况。由于 $\boldsymbol{Ax}=\boldsymbol{b}$ 无解等价于 $\boldsymbol{Hy}=\boldsymbol{d}$ 无解, 此时外包计算者只需返回 $\boldsymbol{y}=\phi$。该结果正确性显而易见。下面对此种情况的隐私性进行说明。由定理 2 证明过程, 系数矩阵 $\boldsymbol{A}$、系数向量 $\boldsymbol{b}$ 的隐私性与 $\boldsymbol{y}$ 的存在性无关。唯一会泄露给外包计算者的信息是: 外包的方程组无解, 即 $\boldsymbol{x}$ 不存在。这一点在实际应用中并不会造成隐私泄露问题。又假设存在 $\boldsymbol{y}^*\neq\phi$ 可以通过验证, 则有 $\boldsymbol{Hy}^*=\boldsymbol{d}$, 与方程 $\boldsymbol{Hy}=\boldsymbol{d}$ 无解矛盾。证毕。综上所述, 当 $\boldsymbol{Ax}=\boldsymbol{b}$ 无解时, 本方案仍具有正确性和足够的隐私性。于是本方案同样适用于处理无解大规模线性方程组问题。

6.2 可审计性和公平支付分析

(1) 可审计性。外包计算者在外包计算的过程中并不一定是诚实的。本文借助区块链公开透明、数据不可篡改的特性, 将区块链作为外包计算的审计平台, 并定义了三类上链记录的外包计算审计事务, 为后续用户提供了外包计算者的审计信息。其中包含的 *wID* 和 *verifyResult*, 可具体标识外包计算者的计算行为: 若 *verifyResult* 为 *True*, 表示此外包计算者在此次计算任务中是诚实的; 若 *verifyResult* 为 *False*, 表示此外包计算者是不诚实的。通过在链上记录的审计事务信息, 可以实现对外包计算者计算行为的公开审计。

(2) 公平支付。在现实生活中外包计算中的实体(主要包括用户和外包计算者)可能并不诚实, 相互之间可能会存在不信任问题。利用区块链智能合约, 在用户通过区块链发布外包计算任务之前, 需要向智能合约支付酬金; 外包计算者通过区块链领取任务时, 需要向智能合约支付押金。如果外包计算任务被正常执行, 用户获得计算结果, 外包计算者将获得用户支付的酬金奖励。若外包计算者存在恶意行为返回错误结果, 其押金将转移给用户作为惩罚。在智能合约按所制定规则自动执行奖惩的机制下, 可以实现用户与外包计算者之间的公平支付。

7 性能分析与方案对比

7.1 时间复杂度分析

在本方案中, 用户的计算开销主要涉及 3 个阶段: KeyGen 阶段, ProbGen 阶段和 Solve 阶段。KeyGen 阶段生成密钥(两个非奇异稀疏矩阵 $\boldsymbol{M}$、$\boldsymbol{N}$)会产生计算开销, 稀疏矩阵每一行非零元素的个数都小于某个 $\rho(1 \ll \rho \ll n)$, 可得此阶段时间复杂度是 $O(\rho n)$; ProbGen 阶段用户计算 $\boldsymbol{d}=\boldsymbol{Mb}$, $\boldsymbol{H}=\boldsymbol{MAN}$, 计算 $\boldsymbol{d}=\boldsymbol{Mb}$ 的计算复杂度是 $O(\rho n)$, 计算 $\boldsymbol{H}=\boldsymbol{MAN}$ 的时间复杂度是 $O(\rho n^2)$, 考虑到针对的是大规模线性方程组的外包, 故 $\rho \ll n$, 所以此阶段计算复杂度为 $O(n^2)$; Solve 阶段用户计算 $\boldsymbol{x}=\boldsymbol{N}(\boldsymbol{y}_1+\boldsymbol{y}_2)$ 会产生计算开销, 此阶段时间复杂度是 $O(\rho n)$。因此本文外包方案中用户计算复杂度为 $O(n^2)$, 而用户通过本地直接求解 $\boldsymbol{Ax}=\boldsymbol{b}$ 的计算复杂度为 $O(n^3)$。所以相比于用户直接对方程组求解, 本文方案能大幅降低用户的计算复杂度。

7.2 方案对比

方案[2, 18, 20, 29]所提出的大规模线性方程组外包计算方案同样采用稀疏矩阵盲化技术, 且这四个方案的实现效率较高。故本文将从用户预计算的计算量、用户验证外包结果的计算量、用户恢复最终结果的计算量以及是否可公开验证这三个方面将本文方案与已有方案[2, 18, 20, 29]进行比较。比较结果如表 1 示, 其中 M 表示数乘, ρ 为所选稀疏矩阵每行(或列)的非零元素的最大数量(是大于 2 且远小于 n 的整数), n 为大规模线性方程组系数矩阵的阶, 是一个密码学中大素数级的整数。

如表 1 可知, 本文方案的算法效率与方案[2, 18, 20, 29]的效率相近, 且与方案[2, 18, 20]相比, 本文方案能实现可公开验证。

7.3 表现评估

从 7.2 节的理论分析可以看出本文方案的优越性, 下面将对外包计算智能合约的开销和外包计算方案的各阶段算法效率进行测试。其中用户和外包方部署在同一台设备上测量算法的运行时间, 这样更能客观的显示用户和外包方的计算量时间比。实验设备配置为 Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 主频的处理器和 8G 内存。

本文使用 Solidity 语言实现了 5.3 节中提出的智能合约, 并在基于浏览器的在线 Remix-Ethereum IDE 上对智能合约的开销进行测试。具体部署环境为: COMPILER(0.4.26+commit4563c3fc), LANGUAGE (Solidity), EVM VERSION(default)测试结果如表 2 所示。

表 1 大规模线性方程组求解外包方案计算性能比较

Table 1 Scheme comparison

方案	ProbGen	Verify	Solve	Public verification
Ref[2]	$((2\rho+1)n^2+\rho n)$M~$O(n^2)$	n^2M~$O(n^2)$	ρnM ~ $O(n)$	×
Ref[18]	$2n^2$M~$O(n^2)$	$2n^2$M~$O(n^2)$	$3n^2$M~$O(n^2)$	×
Ref[20]	$((2\rho+1)n^2+\rho n)$M~$O(n^2)$	$((2\rho+1)n^2)$M~$O(n^2)$	ρnM ~ $O(n)$	×
Ref[29]	$(2n^2+2(\rho-2)n)$M~$O(n^2)$	$3n^2$M~$O(n^2)$	$(n^2+(\rho-2)n)$M~$O(n^2)$	√
Our Scheme	$(2\rho n^2+\rho n)$M~$O(n^2)$	$2n^2$M~$O(n^2)$	ρnM ~ $O(n)$	√

表 2　智能合约开销测试表

Table 2　Smart contract cost test table

智能合约算法	KGAS	GWEI	ETH
Deployment contract	2566	2	0.005132
submitTask	9168	2	0.018336
getTask	126	2	0.000252
uploadResult	28.4	2	0.0000568
verify	290.8	2	0.0005816
receiveResult	381.7	2	0.0007634
terminate	49.7	2	0.0000994

实验以 KGAS(1KGAS=1000GAS)为单位表示消耗, 并设置 GAS LIMIT 为 10000000, GAS 价格为 2GWEI, 相关交易代码在执行过程中会消耗一定量的 GAS, GAS 的消耗值与 GAS 的价格 GWEI 的乘积即为本次交易所花费的以太坊费用。

从表 2 可以看出, 本文方案的 GAS 消耗主要集中在智能合约部署(Deployment contract)和任和任务提交(submitTask)阶段, 分别需要消耗 2566KGAS 和 9168KGAS。相比购买一整套算力设备, 用户通过本文方案进行外包计算产生的手续费在一个可接受的范围。

对于 5.1 节所提大规模线性方程组求解的外包计算方案, 本文使用 MATLAB R2022a 对方案中 ProbGen、Compute、Verify 和 Solve 4 个阶段进行实验测试, 其中 Compute 阶段和 Verify 阶段仅对其计算部分进行实验。表 3 给出了本文方案的性能增益即 $t_{original}/t_{user}$, 可以看出随着维度的扩展, 用户可以获得更多的性能增益。

表 3　大规模线性方程组外包求解实验结果(单位: s)

Table 3　Experiment result

No.	Dimension(n)	$t_{original}$(s)	t_{user}(s)	$t_{worker1}$(s)	$t_{worker2}$(s)	$t_{original}/t_{user}$
1	1000	1.7656	0.2538	1.6604	1.6057	6.96
2	2000	28.3281	3.2228	27.7802	27.6627	8.79
3	3000	115.8125	8.4907	116.232	116.784	13.64
4	4000	299.1563	15.8872	298.165	298.117	18.83
5	5000	613.4375	29.1004	614.653	614.021	21.08
6	6000	1222.1406	45.3821	1222.263	1222.817	26.93
7	7000	1870.2969	61.2409	1871.276	1871.302	30.54
8	8000	3033.2656	88.5525	3032.231	3032.078	34.25

本文方案中用户方总的计算时间包括 ProbGen 阶段和 Solve 阶段, 如图 3 所示, 其中横坐标代表矩阵规模, 纵坐标代表计算时间(单位: s)。可以看出, 本文方案中用户的时间开销低于方案[2, 18, 20, 29]的时间开销, 5 个方案中用户的时间开销均与问题规模呈线性关系。

图 4 比较了用户在不使用外包计算方案和使用本文外包计算方案两种情况下进行大规模线性方程组求解的时间开销。可以看出, 使用本文提出的外包计算方案花费的时间要少的多。

图 5 对外包计算方案中不同阶段的时间开销进行了比较。可以看出, ProbGen 阶段是最耗时的。

8　结论

本文研究了大规模线性方程组的安全外包问题, 利用稀疏矩阵盲化技术、随机分割方法和区块链技术, 构造了一种可审计大规模线性方程组求解外包计算方案。稀疏矩阵盲化技术与随机分割方法使得用户能高效的完成数据的加密、结果的验证与恢复, 保证了用户的数据安全。通过所设计的外包计算智能合约实现外包结果的公开验证和用户与外包计算

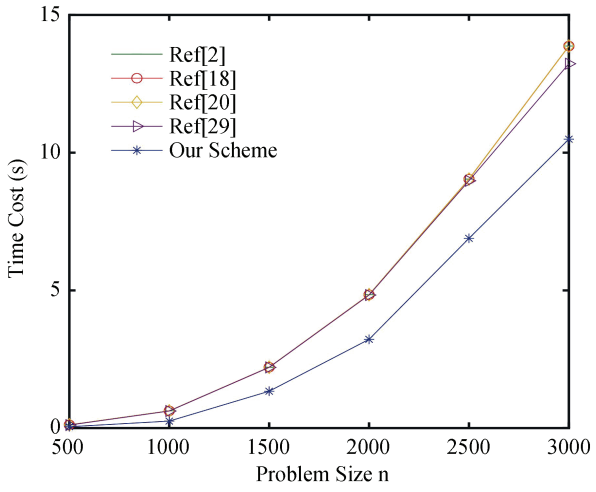

图 3　用户时间开销对比

Figure 3　User-side time cost evaluation and comparison

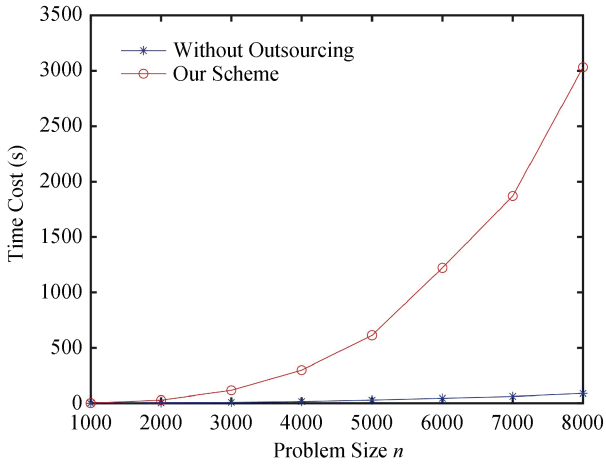

图 4　有无外包计算方案对比

Figure 4　User-side time cost evaluation and comparison

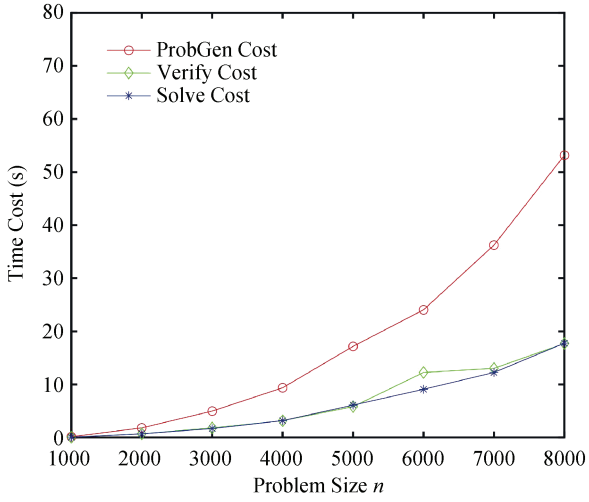

图 5　外包计算方案各阶段时间开销

Figure 5　User-side time cost evaluation and comparison

者之间的公平支付。外包过程中的交互信息上链记录，能实现对外包计算者的审计，并可对恶意外包计算者进行追溯。具有去中心化、不可篡改性及匿名化等特点的区块链技术与可验证外包计算结合，可以在医疗信息系统、机器学习等领域得到很好的具体应用。将区块链与可验证外包计算相结合，解决现存在的人脸识别外包等关键问题，是未来研究工作的重点。

参考文献

[1] Xue R, Wu Y, Liu M H, et al. Progress in Verifiable Computation[J]. *Scientia Sinica* (*Informationis*), 2015, 45(11): 1370-1388.
(薛锐, 吴迎, 刘牧华, 等. 可验证计算研究进展[J]. *中国科学: 信息科学*, 2015, 45(11): 1370-1388.)

[2] Chen X F, Huang X Y, Li J, et al. New Algorithms for Secure Outsourcing of Large-Scale Systems of Linear Equations[J]. *IEEE Transactions on Information Forensics and Security*, 2015, 10(1): 69-78.

[3] Hu X, Pei D Y, Tang C M, et al. Verifiable and Secure Outsourcing of Matrix Calculation and Its Application[J]. *Scientia Sinica* (*Informationis*), 2013, 43(7): 842-852.
(胡杏, 裴定一, 唐春明, 等. 可验证安全外包矩阵计算及其应用[J]. *中国科学: 信息科学*, 2013, 43(7): 842-852.)

[4] Benzi M. Preconditioning Techniques for Large Linear Systems: A Survey[J]. *Journal of Computational Physics*, 2002, 182(2): 418-477.

[5] Joldes G R, Wittek A, Miller K. Real-Time Nonlinear Finite Element Computations on GPU – Application to Neurosurgical Simulation[J]. *Computer Methods in Applied Mechanics and Engineering*, 2010, 199(49/50/51/52): 3305-3314.

[6] Barboteu M, Djehaf N, Martel M. Numerically Accurate Code Synthesis for Gauss Pivoting Method to Solve Linear Systems Coming from Mechanics[J]. *Computers & Mathematics with Applications*, 2019, 77(11): 2883-2893.

[7] Qin C, Wang X B, Zhao N, et al. Research on the Iterative Solver of Linear Equations in Three-Dimensional Finite Element Forward Modeling for Frequency Domain Electromagnetic Method[J]. *Chinese Journal of Geophysics*, 2020, 63(8): 3180-3191.
(秦策, 王绪本, 赵宁, 等. 频率域电磁法三维有限元正演线性方程组迭代算法[J]. *地球物理学报*, 2020, 63(8): 3180-3191.)

[8] Nakamoto S, Bitcoin A. A Peer-To-Peer Electronic Cash System[J]. *Bitcoin*. URL: https://bitcoin. org/bitcoin. pdf, 2008, 4: 2.

[9] Alsberg P A, Day J D. A Principle for Resilient Sharing of Distributed Resources[C]. *Proc International Conference on Software Engineering*, 1976: 562-570.

[10] Canetti R, Riva B, Rothblum G N. Practical Delegation Of Computation Using Multiple Servers[C]. *Proceedings of the 18th ACM conference on Computer and communications security*. 2011: 445-454.

[11] van den Hooff J, Kaashoek M F, Zeldovich N. VerSum: Verifiable Computations over Large Public Logs[C]. *The 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014: 1304-1316.

[12] Belenkiy M, Chase M, Erway C C, et al. Incentivizing Outsourced Computation[C]. *The 3rd international workshop on Economics of networked systems*, 2008: 85-90.

[13] Küpçü A. Incentivized Outsourced Computation Resistant to Malicious Contractors[J]. *IEEE Transactions on Dependable and Secure Computing*, 2017, 14(6): 633-649.

[14] Gentry C. Fully Homomorphic Encryption Using Ideal Lattices[C]. *The forty-first annual ACM symposium on Theory of computing*, 2009: 169-178.

[15] Wang C, Ren K, Wang J, et al. Harnessing the Cloud for Securely Outsourcing Large-Scale Systems of Linear Equations[C]. *IEEE Transactions on Parallel and Distributed Systems*, 2013: 1172-1181.

[16] Paillier P. Public-key Cryptosystems Based On Composite Degree Residuosity Classes[C]. *International Conference on Theory and Application of Cryptographic Techniques*, 1999: 223-238.

[17] Atallah M J, Pantazopoulos K N, Rice J R, et al. Secure Outsourcing of Scientific Computations[J]. *Advances in Computers*, 2002, 54: 215-272.

[18] Cai J X, Ren Y L. Verifiable Outsourcing Algorithm for Large-Scale Systems of Linear Equations[J]. *Application Research of Computers*, 2017, 34(2): 536-538.
(蔡建兴, 任艳丽. 大型线性方程组求解的可验证外包算法[J]. *计算机应用研究*, 2017, 34(2): 536-538.)

[19] Li D M, Dong X L, Cao Z F, et al. Privacy-Preserving Large-Scale Systems of Linear Equations in Outsourcing Storage and Computation[J]. *Science China Information Sciences*, 2018, 61(3): 032112.

[20] Feng D, Zhou F C, Wang Q, et al. Efficient Verifiable Outsourcing of Solving Large-Scale Linear Equations with Low Storage Overhead[J]. *Journal of Computer Research and Development*, 2019, 56(5): 1123-1131.
(冯达, 周福才, 王强, 等. 高效低存储开销可验证外包求解大规模线性方程组方案[J]. *计算机研究与发展*, 2019, 56(5): 1123-1131.)

[21] Golem Network[EB/OL]. https://golem.network/.

[22] Sonm[EB/OL]. https://sonm.com/.

[23] iExec[EB/OL]. https://www.iex.ec/.

[24] Dong C Y, Wang Y L, Aldweesh A, et al. Betrayal, Distrust, and Rationality: Smart Counter-Collusion Contracts for Verifiable Cloud Computing[C]. *The 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017: 211-227.

[25] Huang H, Chen X F, Wu Q H, et al. Bitcoin-Based Fair Payments for Outsourcing Computations of Fog Devices[J]. *Future Generation Computer Systems*, 2018, 78: 850-858.

[26] Krol M, Psaras I. Secure Payments for Outsourced Computations[C]. *2018 NDSS Workshop on Decentralised IoT Security and Standards*, 2018.

[27] Hu S S, Cai C J, Wang Q, et al. Searching an Encrypted Cloud Meets Blockchain: A Decentralized, Reliable and Fair Realization[C]. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018: 792-800.

[28] Lin C, He D B, Huang X Y, et al. Blockchain-Based System for Secure Outsourcing of Bilinear Pairings[J]. *Information Sciences*, 2020, 527: 590-601.

[29] Zhang H L, Gao P, Yu J, et al. Machine Learning on Cloud with Blockchain: A Secure, Verifiable and Fair Approach to Outsource the Linear Regression[J]. *IEEE Transactions on Network Science and Engineering*, 2022, 9(6): 3956-3967.

[30] Guan Y G, Zheng H, Shao J, et al. Fair Outsourcing Polynomial Computation Based on the Blockchain[J]. *IEEE Transactions on Services Computing*, 2022, 15(5): 2795-2808.

[31] Chaum D, Pedersen T P. Wallet Databases with Observers[M]. Chaum D, ed. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007: 89-105.

[32] Benjamin D, Atallah M J. Private and Cheating-Free Outsourcing of Algebraic Computations[C]. *2008 Sixth Annual Conference on Privacy, Security and Trust*, 2008: 240-245.

[33] Gennaro R, Gentry C, Parno B. Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers[M]. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010: 465-482.

[34] Hohenberger S, Lysyanskaya A. How to Securely Outsource Cryptographic Computations[M]. *Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005: 264-282.*

[35] Sun Z X, Zhang X, Xiang F, et al. Survey of Storage Scalability on Blockchain[J]. *Journal of Software*, 2021, 32(1): 1-20.
(孙知信, 张鑫, 相峰, 等. 区块链存储可扩展性研究进展[J]. *软件学报*, 2021, 32(1): 1-20.)

丁艳 于 2021 年在西北民族大学计算机科学与技术专业获得学士学位。现在信息工程大学网络空间安全专业攻读硕士学位。研究领域为云计算安全、可验证计算。研究兴趣包括: 大数据安全、区块链等。Email: 1339437537@qq.com

王娜 于 2008 年在信息工程大学通信与信息系统专业获得博士学位。现任信息工程大学教授。研究领域为云计算安全、网络空间安全。研究兴趣包括: 区块链、信任管理等。Email: twftina_w@126.com

杜学绘 于 2012 年在信息工程大学网络空间安全专业获得博士学位。现任信息工程大学教授。研究领域为大数据安全、云计算安全和信息系统多级安全。研究兴趣包括: 隐私保护、信任管理等。Email: DXH37139@163.com