通用可组合框架下的公平理性委托计算

田有亮[1,2,3]，蒋小霞[1,2,3]

（1. 贵州大学计算机科学与技术学院，贵州 贵阳 550025；2. 贵州大学省部共建公共大数据国家重点实验室（筹），贵州 贵阳 550025；
3. 贵州大学密码学与数据安全研究所，贵州 贵阳 550025）

摘 要：为实现理性委托计算中的经济与信誉公平性，基于密码学的区块链模型提出了通用可组合（UC）框架下的公平理性委托计算方案。首先，结合直接信誉和间接信誉构建关于委托方与计算方的双向信誉激励模型。其次，基于博弈论构建具有完美信息的理性委托计算动态博弈模型，分析得到唯一子博弈纳什均衡。再次，根据理性委托计算场景中的可验证性安全需求、参与者理性决策需求、经济与信誉公平需求以及敌手模型，基于通用可组合理论提出公平理性委托计算理想函数。最后，结合简洁承诺证明和智能合约提出了可安全实现理想函数的公平理性委托计算协议。协议分析证明，所提协议满足 UC 安全性。

关键词：公平理性委托计算；通用可组合框架；智能合约；密码学的区块链模型

中图分类号：TN92

文献标识码：A

DOI: 10.11959/j.issn.1000−436x.2021126

Fair and rational delegation of computation in the universally composable framework

TIAN Youliang[1,2,3], JIANG Xiaoxia[1,2,3]

1. College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
2. State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
3. Institute of Cryptography & Data Security, Guizhou University, Guiyang 550025, China

Abstract: To realize the economic and reputation fairness in rational delegation of computation, a fair and rational delegation of computation scheme based on blockchain model of cryptography in the universally composable (UC) framework was proposed. Firstly, a bidirectional reputation incentive model between the delegator and the calculator was presented by combining direct reputation and indirect reputation. Secondly, a dynamic game model with perfect information for fair and rational delegation of computation was constructed based on game theory, and a unique sub-game Nash equilibrium was further obtained. In addition, by analyzing the requirements for verifiability, rational decision of players, economic and reputation fairness, as well as adversary model, the ideal functionality for fair and rational delegation of computation was proposed. Finally, a fair and rational delegation of computation protocol by combining succinct commit-and-prove and smart contract was proposed to securely realize the ideal functionality. The security analysis shows that the proposed protocol satisfies UC security.

Keywords: fair and rational delegation of computation, universally composable framework, smart contract, blockchain model of cryptography

收稿日期：2021−01−06；**修回日期**：2021−03−29

基金项目：国家自然科学基金资助项目（No.U1836205, No.61662009, No.61772008）；贵州省科技计划基金资助项目（No.20183001, 黔科合基础[2019]1098, 黔科合平台人才[2020]6008, 黔教合人才团队[2013]09）；贵阳市科技计划基金资助项目（筑科合[2021]1-5）

Foundation Items: The National Natural Science Foundation of China (No.U1836205, No.61662009, No.61772008), The Science and Technology Program of Guizhou Province (No.20183001, No.[2019]1098, No.[2020]6008, No.[2013]09), Science and Technology Program of Guiyang (No.[2021]1-5)

1　引言

委托计算[1]是云计算与大数据背景下的新型计算范式，资源受限的委托方可将自己无法完成的计算任务委托给具有强大资源的计算方。针对如何验证计算方返回结果正确性的问题，传统的委托计算通常基于计算复杂度理论构造方案，其主要工具为交互式证明系统[2]、概率检测证明定理[3]、非交互式证明[4-7]等。其中，Costello 等[7]于 2015 年基于简洁承诺证明（SCP, succinct commit-and-prove）提出一个通用的可公开验证计算系统 Geppetto。

近年来，将博弈论与传统委托计算相结合的理性委托计算成为研究热点。Belenkiy 等[8]提出了一个激励外包计算方案，对不同类型的承包商进行奖惩。Kupcu[9]指出文献[8]不提供公平支付，无法抵抗非理性的恶意承包商，因此提出了一个能抵抗恶意承包商的激励外包计算方案。Dong 等[10]将博弈论与智能合约结合，设置不同合约激励计算方诚实执行协议。Tian 等[11]将博弈论与信息论相结合探究了委托计算中参与者的攻防极限。

除了经济激励以外，信誉激励是理性委托计算方案的另一分支。Zhang 等[12]提出了基于信誉的激励众包计算协议，并采用重复博弈框架建立博弈模型。Ma 等[13]利用演化博弈理论构建了可信众包服务中基于信誉的激励博弈模型。Jiang 等[14]提出了通用可组合（UC, universally composable）框架[15]下基于信誉和合同理论的理性委托计算协议。

在以上基于经济或信誉激励的委托计算方案中，文献[8,10,12,14]假设委托方为诚实参与者，仅对理性计算方设置激励机制；文献[9,11,13]虽然将委托方与计算方均视为理性，但是往往假设存在一个可信的第三方系统维护参与者之间的经济支付与信誉评价。为实现公平性，许多学者利用区块链去中心化特性基于比特币[16]设计外包计算公平支付方案[17-19]。

2016 年，Kosba[20]提出了密码学的区块链模型 Hawk，利用 UC 理论描述与区块链交互的分布式协议安全性。同年，Juels 等[21]采用 Hawk 模型分析智能合约的安全性。2020 年，Dorsala 等[22]基于智能合约实现了外包计算的公平支付方案，并采用文献[20-21]中框架分析协议安全性。该方案针对传统基于验证的 1 对 1（即一个委托方对一个计算方）委托计算场景，采用文献[6]中算法实现协议的公开可验证性，但方案中缺乏相应的博弈模型、UC 安全模型以及 UC 安全性证明。

本文利用文献[20-21]中模型探究在 UC 框架下的公平理性委托计算，具体研究工作如下。

1) 构建双向信誉激励模型。本文结合直接信誉和间接信誉形成全局信誉模型，其中，委托方与计算方均具有信誉，双方根据自己的信誉要求选择合适的合作方，且参与者的行为对其信誉产生影响。

2) 基于博弈论构建理性委托计算博弈模型。本文将委托方与计算方均视为理性参与者，根据双方行为策略形式化具有完美信息的动态博弈，通过博弈分析得到委托方与计算方均选择诚实执行协议时达到该博弈模型下的唯一子博弈精炼纳什均衡。

3) 构建公平理性委托计算 UC 安全模型。本文分析公平理性委托计算场景需满足可验证性安全需求、参与者理性决策需求、经济与信誉公平需求，以及敌手模型，基于 UC 理论构造公平理性委托计算理想函数 $\mathcal{F}(\text{Ideal}-\text{FRDC})$。

4) 设计可安全实现理想函数 $\mathcal{F}(\text{Ideal}-\text{FRDC})$ 的公平理性委托计算协议。本文分别采用 SCP 方案[7]实现公开可验证性、利用博弈论分析参与者的理性行为、采用智能合约实现经济与信誉公平性。方案分析证明了该协议满足 UC 安全性。

2　准备知识

2.1　SCP

定义 1　SCP。对于表示计算函数 $F(x)$的所有变量元组 χ，其固定长度为 l，以及对应的多项式时间可验证关系 $\{\Re_\lambda\}_{\lambda\in\mathbb{N}}$，Geppetto 协议[7]包含 6 个多项式时间算法 $(\text{KenGen}=(\text{KenGen}_1,\text{KenGen}_2),\ \text{Digest},\text{Prove},\text{Verify}=(\text{Verify}_1,\text{Verify}_2))$。

密钥产生包含 2 个概率算法。$(\tau)\leftarrow\text{KeyGen}_1(1^\lambda)$：将安全参数 λ 作为输入，输出包含仿真和提取部分的陷门 $\tau=(\tau_S,\tau_E)$，陷门独立于关系 $\mathcal{R}\in\Re_\lambda$。$(\text{EK},\text{VK})\leftarrow\text{KeyGen}_2(\tau,\mathcal{R})$：将陷门 τ 和关系 $\mathcal{R}\in\Re_\lambda$ 作为输入，输出公开评估密钥 EK 和公开验证密钥 VK。其中，EK 包含摘要密钥 EK_z，VK 包含摘要验证密钥 VK_z，$z\in[l]$。

$C_z^{(t)}\leftarrow\text{Digest}(\text{EK}_z,\chi_z^{(t)},o_z^{(t)})$：将摘要密钥 EK_z、消息实例 $\chi_z^{(t)}$ 以及对应的随机数 $o_z^{(t)}$ 作为输入，确定摘要算法产生 $\chi_z^{(t)}$ 的承诺 $C_z^{(t)}$。

$\pi\leftarrow\text{Prove}(\text{EK},\chi,\mathcal{O})$：输入评估密钥 EK，消息 χ 和随机数组 $\mathcal{O}$，确定证明算法输出简洁证明

π，即 π 的长度是关于 λ 的多项式。

$\{0,1\} \leftarrow \text{Verify}_1(\text{VK}_z, C_z^{(t)})$：将摘要验证密钥 VK_z 和摘要 $C_z^{(t)}$ 作为输入；若摘要通过验证，确定摘要验证算法输出 1，否则输出 0。

$\{0,1\} \leftarrow \text{Verify}_2(\text{VK}, \mathcal{C}, \pi)$：将验证密钥 VK、$l$ 个摘要 $\mathcal{C}$ 和证明 π 作为输入；若证明通过验证，确定验证算法输出 1，否则输出 0。

2.2 博弈论

定义 2 扩展式博弈[23]。扩展式博弈的基本表达式为 $G=\{P,A,S,s,Z,U\}$，且参与者的行动具有先后顺序。其中，参与者集合 $P=\{P_1,P_2,\cdots,P_n\}$；行动集合 $A=\{A_1,A_2,\cdots,A_n\}$，$A_i=\{a_i\}$ 为参与者 P_i 可以选择的行动集合；策略集合 $S=\{S_1,S_2,\cdots,S_n\}$，参与者 P_i 可以选择的策略集合用 $S_i=\{s_i\}$ 表示；策略组合 $s=(s_1,s_2,\cdots,s_n)$，其中策略 $s_i \in S_i$；Z 为博弈模型的终端节点集合；支付组合 $U=(u_1,u_2,\cdots,u_n)$，其中，$u_i: Z \to \mathbb{R}$ 表示参与者 P_i 在终端节点上的效用函数[10]。

定义 3 子博弈精炼纳什均衡[23]。在一个具有完美信息的动态博弈中，如果满足以下条件，则各参与者的策略组合 $s^*=(s_1^*,\cdots,s_i^*,\cdots,s_n^*)$ 是一个子博弈精炼纳什均衡。

1) 它是原博弈的纳什均衡。

2) 它在每一个子博弈中都构成纳什均衡。

2.3 UC 框架

定义 4 UC 仿真[15]。令 Π 和 ρ 为概率多项式时间（PPT, probabilistic polynomial time）协议。如果对于任何 PPT 敌手 $\mathcal{A}$，存在 PPT 敌手 $\mathcal{S}$ 使对任何 PPT 环境 $\mathcal{Z}$，无法以一个不可忽略的概率区分现实世界与理想世界，在现实世界中 $\mathcal{Z}$ 与 $\mathcal{A}$、Π 交互，而在现实世界中 $\mathcal{Z}$ 与 $\mathcal{S}$、$\mathcal{F}$ 交互，则称协议 Π UC-仿真协议 ρ。

定义 5 UC 实现[15]。令 $\mathcal{F}$、Π 分别为理想函数和协议。如果协议 Π UC-仿真 $\mathcal{F}$ 的理想协议，则称协议 Π UC-实现理想函数 $\mathcal{F}$，也称协议 Π 安全实现理想函数 $\mathcal{F}$，此时 Π 是 UC 安全的。

2.4 智能合约

智能合约[24]是运行在区块链上的一个计算机程序实例，由所有共识节点执行。其主要由程序代码、存储文件和账户余额组成。任何用户都可以通过向区块链发送事务来创建合约，然后将合约部署到区块链中。合约创建时，固定的程序代码反映了交易用户间的合约条款逻辑，一旦满足合约触发条件便自动执行，不受外界干预。因此，可将智能合约看作一个可信的第三方，其代码逻辑执行的正确性由共识机制保证。

2.5 密码学的区块链模型

密码学的区块链模型[20]将理想描述以及在区块链上或者用户端运行的协议用伪码形式表示成程序，分别被称为理想程序、合约程序和用户程序。该模型依赖封装器对协议进行模块化设计，可将程序编译成 UC 框架中的函数或者协议。具体地，理想封装器 $\mathcal{F}(\cdot)$ 将理想程序转换成理想函数 $\mathcal{F}(\text{Ideal}P)$；合约封装器 $\mathcal{G}(\cdot)$ 将合约程序转换成合约函数 $\mathcal{G}(\text{Con}P)$，此合约函数包含在区块链上执行的程序；协议封装器 $\Pi(\cdot)$ 将用户程序转换为用户端协议 $\Pi(\text{User}P)$。

3 问题陈述

3.1 系统模型

本文假设场景中存在 M 个委托方与 N 个计算方，则委托方集合与计算方集合分别可用 $\mathbb{D}=\{D_1,\cdots,D_M\}$ 和 $\mathbb{W}=\{W_1,\cdots,W_N\}$ 表示。系统模型如图 1 所示，具体描述如下。

首先，委托方 D_i 需向计算方集合广播任务请求及信誉要求，其中，$\forall i \in \{1,2,\cdots,M\}$。一旦收到来自计算方的响应消息，便从区块链查看响应者的历史交互评价并计算其信誉值，选择信誉最高且满足自己信誉要求的计算方签署委托合约。D_i 发送相关信息触发委托合约，并私下将真实的任务发送给被选择的计算方。一旦收到来自计算方的计算结果，D_i 产生结果摘要并分别发送至智能合约和计算方。当从计算方处收到中间参数摘要和结果证明后，D_i 在本地验证摘要和证明，然后向智能合约发送确认消息以表明计算方成功执行任务。

计算方 W_j 一旦收到 D_i 的广播信息，便在本地计算 D_i 的信誉，其中，$\forall j \in \{1,2,\cdots,N\}$。一旦 D_i 的信誉满足自己的信誉要求，便向 D_i 发送响应信息。若 W_j 被成功委任，则在本地计算来自 D_i 的任务，然后向智能合约提交结果承诺和中间参数摘要，并在私下将结果和中间参数摘要发送给 D_i。收到来自 D_i 的结果摘要后，W_j 生成结果证明并将其私下发给 D_i，将结果证明承诺发送给智能合约。若 W_j 在规定的时间内未收到合理的支付，可向智能合约发

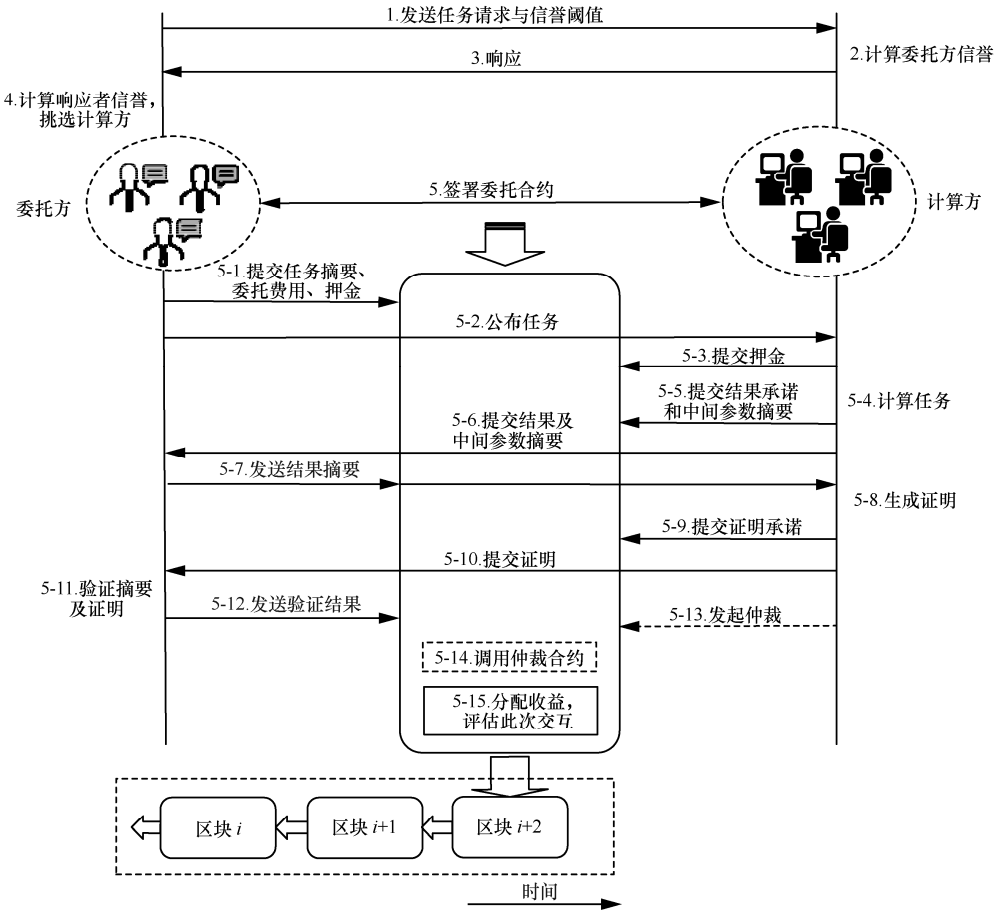

图 1　系统模型

起仲裁请求。

一旦收到来自委托方的确认消息，智能合约仅需负责相应的收益分配和交互评价。但当收到来自计算方的仲裁请求时，智能合约还将调用内部的仲裁合约解决争端。

由于仅当 D_i 选择不诚实行为时，W_j 才有可能向委托合约发起仲裁请求，委托合约随之调用仲裁合约，因此，本文在系统模型图中将这 2 个步骤用虚线表示以区别于正常的交互流程。本文将证明当 D_i 和 W_j 都是理性参与者时，协议将不会执行这 2 个步骤。

3.2　敌手模型

场景中的委托方与计算方均被视为理性参与者，因此两者均存在自利行为从而做出偏离协议的决策。本文考虑以下 2 种攻击策略，且设计的协议在这些可能的攻击行为下仍是安全的。

错误报告。收到来自计算方提交的诚实计算结果后，委托方向智能合约发送认证失败的反馈信息，试图获取因为拥有正确计算结果带来的收益而拒绝向计算方支付委托费用以实现更大效用。

搭便车。收到来自委托方的计算任务后，计算方为了节省计算资源，随意地向智能合约和计算方提交计算结果和证明而非认真执行计算任务。

4　信誉模型

现有大多数信誉方案仅考虑委托方对计算方进行评估的单向信誉模型，计算方不掌握委托方的信誉信息从而无法选择优质的合作者。而本文考虑双向信誉评估，且结合直接信誉和间接信誉最终形成全局信誉模型。下文以委托方对计算方的信誉评估为例，计算方可按照相同方式对委托方进行信誉评估。

4.1　直接信誉

直接信誉是指由参与交易的 D_i 和 W_j 以往相互合作累积形成的交互评价。直接信誉与历史交互相关，且越接近此次交易的交互评价应越能反映参与者当前的信誉。由于委托方与计算方的每次交易及交互评价最终都会部署在区块链上，用参数 t 表示

部署在区块链上的第 t 次交易，且 $t \in \{0,1,2,\cdots\}$，其中 $t=0$ 表示参与者尚未开始交易。

令 $r_{i,j}^t$ 为 W_j 在第 t 次交易中 D_i 交互后获得的交互评价，为了对 W_j 的行为进行奖惩信誉激励，$r_{i,j}^t$ 可设置[13-14]为

$$r_{i,j}^t = \begin{cases} 1, & \mathrm{Jud}_{i,j}^t = \mathrm{positive} \\ -2, & \mathrm{Jud}_{i,j}^t = \mathrm{negative} \\ 0, & \mathrm{Jud}_{i,j}^t = \mathrm{null} \end{cases} \tag{1}$$

其中，当 W_j 在第 t 次交易中诚实执行由 D_i 委任的任务时，$\mathrm{Jud}_{i,j}^t = \mathrm{positive}$；否则，$\mathrm{Jud}_{i,j}^t = \mathrm{negative}$。若在第 t 次交易中 W_j 与 D_i 并未发生交易，则 $r_{i,j}^t = 0$。显然，$r_{i,j}^0 = 0$，$\forall i \in [1,\cdots,M]$，$\forall j \in [1,\cdots,N]$。

令 $x_{i,j}^t$ 为 W_j 在第 t 次交易及以前与 D_i 交互后获得的累积交互评价，且要求距离现在交易越久远的交互评价应对当前信誉产生影响越小，则 $x_{i,j}^t$ 为[25]

$$x_{i,j}^t = \sum_{t'=0}^{t} \lambda^{t-t'} r_{i,j}^{t'} \tag{2}$$

其中，$\lambda \in [0,1]$ 为交互评价的影响因子，且 $\lambda^{t-t'}$ 减小了历史交互评价的影响程度。同样，$x_{i,j}^0 = 0$，$\forall i \in [1,\cdots,M]$，$\forall j \in [1,\cdots,N]$。由于人类交互的信誉评价符合 Gompertz 模型[26]描述的一般发展规律，因此本文采用 Gompertz 模型来构造信誉模型。则有

$$R_{i,j}^{t,\mathrm{dir}}(x_{i,j}^t) = a\mathrm{e}^{b\mathrm{e}^{cx_{i,j}^t}} \tag{3}$$

其中，a、b、c 为模型参数，a 指定信誉的最大值，b 控制沿 x 轴的位移，c 调节信誉模型的增长速率。因此，第 t 次交易后 D_i 对 W_j 当前的直接信誉评估可用 $R_{i,j}^{t,\mathrm{dir}}$ 表示。当 $t=0$ 时，即任何委托方与计算方尚未开始交互，D_i 无法根据与 W_j 以往交互的历史信息对其进行信誉评估，此时 $x_{i,j}^0 = 0$，相应的信誉评估为 $R_{i,j}^{0,\mathrm{dir}}(x_{i,j}^0) = a\mathrm{e}^b$。因此，参与者之间的默认直接信誉评估值便为 $a\mathrm{e}^b$。

4.2 间接信誉

间接信誉是指由其余委托方对计算方产生的累加直接信誉评估，且不同委托方应具有不同权重。此时，其余委托方被视为推荐者，且若 D_k 与 D_i 拥有越多相似的交互历史，即与相同的计算方交易，则其对间接信誉的影响越大。令 $\mathrm{sim}^t(i,k)$ 表示在第 t 次交易后 D_i 与 D_k 的相似系数，本文采用修正余弦函数来衡量相似度，具体表达式为[27-28]

$$\mathrm{sim}^t(i,k) = \frac{\sum_{j\in\mathbb{C}} \left(R_{i,j}^{t,\mathrm{dir}} - \overline{R_i^{t,\mathrm{dir}}}\right)\left(R_{k,j}^{t} - \overline{R_k^{t,\mathrm{dir}}}\right)}{\sqrt{\sum_{j\in\mathbb{I}} \left(R_{i,j}^{t,\mathrm{dir}} - \overline{R_i^{t,\mathrm{dir}}}\right)^2} \sqrt{\sum_{j\in\mathbb{K}} \left(R_{k,j}^{t,\mathrm{dir}} - \overline{R_k^{t,\mathrm{dir}}}\right)^2}} \tag{4}$$

其中，$\mathbb{I}$ 与 $\mathbb{K}$ 分别表示与 D_i 和 D_k 交互过的计算方集合；$\mathbb{C}$ 表示同时与 D_i 和 D_k 交互过的计算方集合，即 $\mathbb{C} = \mathbb{I} \cap \mathbb{K}$；$\overline{R_i^{t,\mathrm{dir}}}$ 与 $\overline{R_k^{t,\mathrm{dir}}}$ 分别表示在第 t 次交易后 D_i 和 D_k 对所有交互过的计算方的平均直接信誉评估；$R_{i,j}^{t,\mathrm{dir}}$ 与 $R_{k,j}^{t,\mathrm{dir}}$ 分别表示在第 t 次交易后 D_i 和 D_k 对 W_j 的直接信誉评估。

所有推荐者对 W_j 的间接信誉评估为

$$R_{i,j}^{t,\mathrm{rec}} = \frac{\sum_{k\in\mathbb{K}} \mathrm{sim}_{i,x}^t R_{k,j}^{t,\mathrm{dir}}}{\sum_{k\in\mathbb{K}} \mathrm{sim}_{i,x}^t} \tag{5}$$

其中，$k \in \mathbb{K}$，$\mathbb{K}$ 表示与 W_j 交互过的推荐者集合。

4.3 全局信誉

参与者最终的信誉被称为全局信誉，包含直接信誉和间接信誉的加权影响。在第 t 次交易后，D_i 对 W_j 的全局信誉评估方式为

$$R_{i,j}^{t,\mathrm{final}} = (1-\alpha)R_{i,j}^{t,\mathrm{dir}} + \alpha R_{i,j}^{t,\mathrm{rec}} \tag{6}$$

其中，$\alpha \in [0,1]$ 为间接影响因子，当 $\alpha \to 1$ 时，间接信誉即其余委托方对 W_j 的信誉评估更能够决定 W_j 的最终信誉，反之亦然。

若 D_i 与 W_j 在第 $t+1$ 次交易期间开始交互，双方将根据对方前 t 次交易的累积全局信誉判断其是否满足自己的最低信誉要求。分别用 $R_{i,\mathrm{th}}^t$ 与 $R_{j,\mathrm{th}}^t$ 表示 D_i 和 W_j 在第 t 次交易时的信誉阈值要求，对方的信誉必须不小于此信誉阈值才会考虑进行合作。

5 博弈模型

传统委托计算仅侧重于方案的可验证性，无法考虑参与者自利行为对协议造成的影响。本文从博弈论角度出发，将委托方和计算方均视为理性参与者，且参与者为理性是共同知识，利用扩展式博弈对参与者进行理性建模，博弈模型如图 2 所示。

此博弈为具有完美信息的动态博弈。D_i 收到来自 W_j 提交的计算结果，并对其进行验证以获知对方是否选择诚实执行任务后，再决定自己的行动。W_j 观测 D_i 是否完成支付的行动后，再选择自己是否发起仲裁请求。

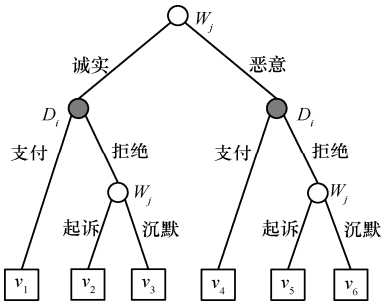

图 2　博弈模型

D_i 与 W_j 在第 t 次交易中的博弈模型可表示为 $G_t=(P,A,S,s,Z,U,E)$。

1) 参与者集合 P 指参与协议的委托方与计算方，$P=\{W_j,D_i\}$。

2) 行动集合 A 指参与者可能选择的所有行动，$A=\{A_j,A_i\}$。W_j 在面对被委托的任务时可能诚实执行，也可能恶意地提交一个错误结果；W_j 在面对 D_i 是否支付委托费用的行动后，可选择是否向智能合约发起仲裁请求。因此，其行动空间为 $A_j=\{$诚实,恶意,起诉,沉默$\}$。而 D_i 在面对由 W_j 提交的计算结果时，可选择向其支付委托费用或者拒绝支付，此时 D_i 的行动空间为 $A_i=\{$支付,拒绝$\}$。

3) 策略集合 S 指参与者所有可选择的策略，$S=\{S_j,S_i\}$。计算方 W_j 作为先行动者仅有 2 个策略，即 $S_{j_1}=\{$诚实,恶意$\}$；而委托方 D_i 的策略集为 $S_i=\{\{$支付,支付$\}$，$\{$支付,拒绝$\}$，$\{$拒绝,支付$\}$，$\{$拒绝，拒绝$\}\}$；此时，计算方 W_j 可进一步选择的策略空间为 $S_{j_2}=\{\{$起诉,起诉$\}$，$\{$起诉,沉默$\}$，$\{$沉默,起诉$\}$，$\{$沉默,沉默$\}\}$。

4) 策略组合 s 指各参与者选择策略形成的二维向量，$s=(s_j,s_i)$。s_j 与 s_i 表示 W_j 和 D_i 选择的特定策略。

5) 终端节点集合 Z 指博弈模型的所有终端节点，$Z=\{v_1,v_2,v_3,v_4,v_5,v_6\}$。

6) 支付组合 $U=(u_j,u_i)$，其中，$u_i,u_j: Z\to\mathbb{R}$ 分别为参与者 W_j 与 D_i 在终端节点 Z 的实值效用函数。

7) 均衡 E 指各参与者的最优策略组合，即 $E=s^*=(s_j^*,s_i^*)$。

D_i 与 W_j 分别需要提前提交 d_i 和 d_j，参与者仅在诚实执行协议时能够被退还押金，否则，参与者将失去押金。诚实的参与者不仅能够收到自己的押金，还将收到非诚实参与者的押金，作为双方在交易中诚实行为的保障。D_i 需向 W_j 支付 w_{ij} 作为诚实完成计算任务的酬劳，而计算方为执行计算任务需要花费 c_j，假设计算方提交错误结果的开销为 0。D_i 需对 W_j 提交的计算结果进行验证，从而花费验证 c_i；而当 D_i 收到诚实计算的结果后将获得 f_i。若 W_j 选择发起仲裁请求，其开销为 h_j。博弈模型中的货币参数及其含义如表 1 所示。

表 1　　货币参数及其含义

参数	含义
w_{ij}	委托方 D_i 向计算方 W_j 支付的委托费用
d_i	委托方 D_i 提交的押金
d_j	计算方 W_j 提交的押金
c_i	委托方 D_i 的验证成本
c_j	计算方 W_j 的计算成本
f_i	委托方 D_i 的收益
h_j	计算方 W_j 支付的仲裁费用

为确保 D_i 与 W_j 之间满足激励相容，货币参数需满足以下关系。

1) $w_{ij}>c_j$。D_i 的委托费用应比 W_j 诚实执行任务的开销大，否则 W_j 不会愿意接受此计算任务。

2) $f_i>w_{ij}+c_i$。D_i 因计算任务获得的收益应比自己付出的委托费用和验证开销总和多，否则 D_i 不会愿意将此计算任务外包给 W_j。

3) $d_i>c_j+h_j$。为确保诚实 W_j 的利益，D_i 的押金应大于 W_j 诚实计算的开销与仲裁费用的总和。

4) $d_j>c_i$。为确保诚实 D_i 的利益，W_j 的押金应大于 D_i 的验证开销。

通过对理性委托计算博弈模型进行分析，得出博弈模型中各终端节点对应委托方与计算方的效用，具体如表 2 所示。

表 2 **效用分析**

终端节点/参与者	W_j	D_i
v_1	$w_{ij}-c_j$	$f_i-w_{ij}-c_i$
v_2	$d_i-c_j-h_j$	$f_i-d_i-c_i$
v_3	$-c_j$	f_i-c_i
v_4	w_{ij}	$-w_{ij}-c_i$
v_5	$-h_j-d_j$	d_j-c_i
v_6	$-d_j$	d_j-c_i

6 理想世界

本节将根据文献[20-21]中的框架描述公平理性委托计算理想程序 Ideal – FRDC。令 $\mathcal{S}$ 为理想敌手。令 $\mathcal{W}_{\text{init}}=\{W_1,\cdots,W_N\}$ 表示包含所有计算方的集合。令 $\mathcal{W}_{\text{res}}$ 表示响应广播的计算方集合，初始化为 $\mathcal{W}_{\text{res}}=\varnothing$。令 $\mathcal{W}_{\text{rep}}$ 表示响应广播且信誉满足信誉要求的计算方集合，初始化为 $\mathcal{W}_{\text{rep}}=\varnothing$。公平理性委托计算理想程序 Ideal – FRDC 的设计如下。

Init

设置 $t:=1$，state:=INIT，$\$\text{reward}:=0$，$\$\text{deposit}:=\{\}$，$\mathcal{W}_{\text{init}}:=\{W_1,\cdots,W_N\}$，$\mathcal{W}_{\text{res}}:=\{\}$，$\mathcal{W}_{rep}:=\{\}$。

Broadcast

当从 D_i 收到 $(\text{Broadcast},\text{sid},\mathcal{R},R_{i,\text{th}}^{t-1})$ 时，将消息 $(\text{Broadcast},\text{sid},\mathcal{R},R_{i,\text{th}}^{t-1},R_{j,i}^{t-1,\text{final}})$ 发送给所有的 $W_j\in\mathcal{W}_{\text{init}}$。

Response

当从 W_j 收到 (Response,sid) 时，执行以下步骤。

1) 将 $(\text{Response},\text{sid},R_{i,j}^{t-1,\text{final}})$ 发送给 D_i。

2) 设置 $\mathcal{W}_{\text{res}}:=\mathcal{W}_{\text{res}}\cup W_j$。

3) 若 $R_{i,j}^{t-1,\text{final}}\geqslant R_{i,\text{th}}^{t-1}$，设置 $\mathcal{W}_{\text{rep}}:=\mathcal{W}_{\text{rep}}\cup W_j$。

4) 将 $(\text{Response},\text{sid},D_i,W_j)$ 分别发送给 D_i 与 W_j，其中 W_j 为 $\mathcal{W}_{\text{rep}}$ 中信誉最高者。

Corrupt

当从 $\mathcal{S}$ 收到消息 $(\text{Corrupt},\text{sid},D_i/W_j)$ 时，记录 $(\text{Corrupt},\text{sid},D_i/W_j)$。

Create

当从 D_i 收到消息 $(\text{Create},\text{sid},x,\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_e,\$w_{ij},\$d_i)$ 时，执行以下步骤。

1) 将消息 $(\text{Create},\text{sid},|x|,\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_e,\$w_{ij},\$d_i,D_i)$ 转发给敌手 $\mathcal{S}$。

2) 假设 state:=INIT。

3) 假设 $\upsilon<\upsilon_1<\upsilon_2<\upsilon_3<\upsilon_4<\upsilon_e$。

4) 假设 $\text{ledger}[D_i]\geqslant\$w_{ij}+\$d_i$。

5) 设置 $\text{ledger}[D_i]:=\text{ledger}[D_i]-(\$w_{ij}+\$d_i)$。

6) 设置 $\$\text{reward}:=\$w_{ij}$。

7) 设置 $\$\text{deposit}:=\$\text{deposit}\cup(\$d_i,D_i)$。

8) 设置 state:=CREATED。

Intent

当从 W_j 收到 $(\text{Intent},\text{sid},\$d_j)$，执行以下步骤。

1) 将消息 $(\text{Intent},\text{sid},\$d_j,W_j)$ 转发给敌手 $\mathcal{S}$。

2) 假设 state:=CREATED。

3) 假设 $\upsilon\leqslant\upsilon_1$。

4) 假设 $\text{ledger}[W_j]\geqslant\$d_j$。

5) 设置 $\text{ledger}[W_j]:=\text{ledger}[W_j]-\$d_j$。

6) 设置 $\$\text{deposit}:=\$\text{deposit}\cup(\$d_j,W_j)$。

7) 设置 state:=INTENT。

Compute

当从 W_j 收到 (Compute,sid)，执行以下步骤。

1) 将消息 $(\text{Compute},\text{sid},W_j)$ 转发给敌手 $\mathcal{S}$。

2) 将 $(\text{Compute},\text{sid},y,W_j)$ 转发给 D_i。

3) 假设 state:=INTENT。

4) 假设 $\upsilon\leqslant\upsilon_2$。

5) 设置 state:=COMPUTED。

Prove

当从 W_j 收到消息 (Prove,sid)，执行以下步骤。

1) 将消息 $(\text{Prove},\text{sid},W_j)$ 转发给敌手 $\mathcal{S}$。

2) 将 $(\text{Prove},\text{sid},\pi,W_j)$ 转发给 D_i。

3) 假设 state:=COMPUTED。

4) 假设 $\upsilon\leqslant\upsilon_3$。

5) 设置 state:=PROVED。

Verify

当从 D_i 收到消息 (Verify,sid)，执行以下步骤。

1) 将消息 $(\text{Verify},\text{sid},D_i)$ 转发给敌手 $\mathcal{S}$。

2) 将 $(\text{Verify},\text{sid},1,D_i)$ 转发给 W_j。

3) 假设 state:=PROVED。

4) 假设 $\upsilon\leqslant\upsilon_4$。

5) 设置 $\text{ledger}[D_i]:=\text{ledger}[D_i]+(\$d_i,D_i)$。

6) 设置 $\text{ledger}[W_j]:=\text{ledger}[W_j]+(\$d_j,W_j)+$

$reward。

7) 设置 $\mathrm{Jud}_{i,j}^{t}$:= positive，$\mathrm{Jud}_{j,i}^{t}$:= positive，存储 $\mathrm{Jud}_{i,j}^{t}$，$\mathrm{Jud}_{j,i}^{t}$。

8) 设置 state:=ABORTED。

与经典的理想函数形式化方法不同，本文基于文献[20-21]中的框架利用理想封装器 $\mathcal{F}(\cdot)$ 将公平理性委托计算理想程序 Ideal – FRDC 转换成公平理性委托计算理想函数 $\mathcal{F}(\text{Ideal} - \text{FRDC})$。在理想世界中，协议参与者即委托方和计算方均被视为愚蠢参与者，任何操作均由理想函数执行，而愚蠢参与者只负责消息的转发。理想函数的执行应满足理性委托计算的传统安全需求、理性需求、公平需求与敌手模型。具体来说，传统安全需求包含计算方提交结果的可验证性；理性需求包含参与者的理性决策；公平需求包含委托方与计算方交互的经济公平性和信誉公平性；敌手模型刻画委托方或计算方可能的攻击行为。

可验证性。计算方提交的结果可被正确验证。对于任意计算结果 y，均存在一个对应的证明 π 可以验证 y 是否正确。

理性决策。参与者均作为自利者，基于理性思考后做出效用最大化决策。

经济公平性。委托方向计算方支付委托费用当且仅当计算方提交的结果被验证为正确。

信誉公平性。参与者被评估为积极交互仅当自己在本次交互中选择诚实行为，否则被记录为消极交互。

攻击行为。被贿赂的委托方将拒绝向计算方支付委托费用；被贿赂的计算方将提交错误的计算结果。

7　现实世界

为安全实现理想世界中的理想函数，现实世界中的协议设计应从可验证安全需求、理性需求和公平需求出发。本文分别采用文献[7]中的简洁承诺证明协议 Geppetto 实现方案的可验证性，基于博弈论分析参与者的理性决策，利用智能合约实现参与者之间的信誉公平性和经济公平性。基于文献[20-21]中的模型，将协议模块化设计为在区块链上运行的委托合约程序 ConP – Delegate 和仲裁合约程序 ConP – Arbitrate，以及在用户端运行的程序 UserP – FRDC。合约封装器 $\mathcal{G}(\cdot)$ 可分别将委托合约程序和仲裁合约程序转换成委托合约函数 $\mathcal{G}(\text{ConP} - \text{Delegate})$ 和仲裁合约函数 $\mathcal{G}(\text{ConP} - \text{Arbitrate})$；协议封装器 $\Pi(\cdot)$ 将用户程序转换为用户端协议 $\Pi(\text{UserP} - \text{FRDC})$。

7.1　合约设计

本文借鉴文献[22]中的思想设计合约，委托方需准备委托合约和仲裁合约。其中，委托合约用于描述委托方与计算方之间的正常交互；仲裁合约用于解决委托方与计算方之间的争端，仅当参与者双方存在争议时才会被触发。

7.1.1　委托合约

当委托方 D_i 与计算方 W_j 均同意执行任务 $F(\cdot)$ 时，双方签署委托合约。当且仅当计算方提交的结果被验证为正确时，委托方才会向计算方支付委托费用 w_{ij}。为激励双方能够诚实行事，要求委托方与计算方均提前提交押金，诚实参与者可重新获得返回的押金，而具有不诚实行为的参与者将失去押金，且此押金将被诚实参与者持有。委托合约的具体描述如下。

1) 委托合约由 D_i 与 W_j 签署，表明 D_i 选择将任务 $F(\cdot)$ 委托给 W_j，且 W_j 也同意完成此任务。

2) D_i 与 W_j 均同意时间参数 $\upsilon < \upsilon_1 < \upsilon_2 < \upsilon_3 < \upsilon_4 < \upsilon_e$，其中，$\upsilon$ 为当前的时间。

3) D_i 需提前提交委托费用 $\$w_{ij}$ 和押金 $\$d_i$，且将任务承诺 $(C_x, H_{\mathrm{EK}}, H_{\mathrm{VK}})$ 作为输入触发委托合约。

4) W_j 需在截止时间 υ_1 之前提交押金 $\$d_j$，否则，合约终止且将押金 $\$d_i$ 和委托费用 $\$w_{ij}$ 退回给 D_i。

5) 在截止时间 υ_2 之前，W_j 需提交计算结果承诺与中间参数摘要 $(H_y, \mathcal{C} \setminus (C_y \cup C_x))$，而后 D_i 需提交摘要 C_y；否则，合约终止且将押金 $\$d_i$、$\$d_j$ 以及委托费用 $\$w_{ij}$ 退回给 D_i。

6) W_j 需在截止时间 υ_3 之前提交证明承诺 H_π。

7) D_i 需在截止时间 υ_4 之前提交验证结果 $\mathrm{Ver}_{i,j}^{t}$。当 $\mathrm{Ver}_{i,j}^{t}$=1 时，合约终止且将 $\$d_j$ 与 $\$w_{ij}$ 发送给 W_j，且 D_i 获得 $\$d_i$。

8) 当 D_i 未在 υ_4 前发送验证结果，或在规定时间内提交的结果不符合计算方 W_j 的预期，W_j 可以选择发送任务与计算结果触发仲裁合约以解决争端，且合约终止。W_j 一旦发起仲裁便花费 $\$h_j$。若仲裁合约返回的结果为 Arb_j^t=1，将 $\$w_{ij}$ 发送给 D_i，

将 $\$d_i$ 与 $\$d_j$ 发送给 W_j。否则，将 $\$d_i$、$\$d_j$ 与 $\$w_{ij}$ 发送给 D_i。

9) 当 $\upsilon > \upsilon_e$ 且合约还未终止，即仲裁合约未能在规定时间内给出仲裁结果时，将 $\$w_{ij}$ 发送给 D_i，将 $\$d_i$ 与 $\$d_j$ 发送给 W_j。

10) 委托合约终止前需对 D_i 与 W_j 的交互行为进行评估，且将 $\mathrm{Jud}_{i,j}^t$ 和 $\mathrm{Jud}_{j,i}^t$ 进行存储。

根据委托合约的设计可基于文献[20-21]的框架形式化描述委托合约程序 ConP – Delegate，具体内容如下。

Init

设置 state:=INIT，$\mathrm{Jud}_{i,j}^t := \perp$，$\mathrm{Jud}_{j,i}^t := \perp$，$\$\mathrm{reward} := 0$，$\$\mathrm{deposit} := \{\}$。

Create

当从 D_i 收到消息 $(\mathrm{Create}, \mathrm{sid}, C_x, H_{\mathrm{EK}}, H_{\mathrm{VK}}, \upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4, \upsilon_e, \$w_{ij}, \$d_i)$，执行以下步骤。

1) 假设 state:=INIT。

2) 假设 $\upsilon < \upsilon_1 < \upsilon_2 < \upsilon_3 < \upsilon_4 < \upsilon_e$。

3) 假设 $\mathrm{ledger}[D_i] \geqslant \$w_{ij} + \$d_i$。

4) 设置 $\mathrm{ledger}[D_i] := \mathrm{ledger}[D_i] - (\$w_{ij} + \$d_i)$。

5) 设置 $\$\mathrm{reward} := \$w_{ij}$。

6) 设置 $\$\mathrm{deposit} := \$\mathrm{deposit} \cup (\$d_i, D_i)$。

7) 设置 state:=CREATED。

Intent

当从 W_j 收到 $(\mathrm{Intent}, \mathrm{sid}, \$d_j)$，执行以下步骤。

1) 假设 state:=CREATED。

2) 假设 $\upsilon \leqslant \upsilon_1$。

3) 假设 $\mathrm{ledger}[W_j] \geqslant \$d_j$。

4) 设置 $\mathrm{ledger}[W_j] := \mathrm{ledger}[W_j] - \$d_j$。

5) 设置 $\$\mathrm{deposit} := \$\mathrm{deposit} \cup (\$d_j, W_j)$。

6) 设置 state := INTENT。

Compute

当从 W_j 收到消息 $(\mathrm{Compute}, \mathrm{sid}, H_y, \mathcal{C} \setminus (C_y \cup C_x))$，从 D_i 收到消息 $(\mathrm{Compute}, \mathrm{sid}, C_y)$，执行以下步骤。

1) 假设 state:=INTENT。

2) 假设 $\upsilon \leqslant \upsilon_2$。

3) 设置 state:=COMPUTED。

Prove

当从 W_j 收到消息 $(\mathrm{Prove}, \mathrm{sid}, H_\pi)$，执行以下步骤。

1) 假设 state:=COMPUTED。

2) 假设 $\upsilon \leqslant \upsilon_3$。

3) 设置 state := PROVED。

Verify

当从 D_i 收到消息 (Verify,sid,1)，执行以下步骤。

1) 假设 state:=PROVED。

2) 假设 $\upsilon \leqslant \upsilon_4$。

3) 设置 $\mathrm{ledger}[D_i] := \mathrm{ledger}[D_i] + (\$d_i, D_i)$。

4) 设置 $\mathrm{ledger}[W_j] := \mathrm{ledger}[W_j] + (\$d_j, W_j) + \$\mathrm{reward}$。

5) 设置 $\mathrm{Jud}_{i,j}^t := \mathrm{positive}$，$\mathrm{Jud}_{j,i}^t := \mathrm{positive}$，存储 $\mathrm{Jud}_{i,j}^t$ 和 $\mathrm{Jud}_{j,i}^t$。

6) 设置 state:=ABORTED。

Prosecute

当从 W_j 收到消息 $(\mathrm{Prosecute}, \mathrm{sid}, \mathrm{EK}, \mathrm{VK}, x, y, \pi, \$h_j)$，执行以下步骤。

1) 假设 $\mathrm{state} != \mathrm{ABORTED}$。

2) 假设 $\upsilon_4 < \upsilon \leqslant \upsilon_e$。

3) 假设 $\mathrm{ledger}[W_j] \geqslant \$h_j$。

4) 假设 $H_{\mathrm{EK}} = H(\mathrm{EK})$，$H_{\mathrm{VK}} = H(\mathrm{VK})$，$H_\pi = H(\pi)$，$C_x = \mathrm{Digest}(\mathrm{EK}_x, x, o_x)$，$C_y = \mathrm{Digest}(\mathrm{EK}_y, y, o_y)$。

5) 设置 $\mathrm{ledger}[W_j] = \mathrm{ledger}[W_j] - \$h_j$。

6) 将消息 $(\mathrm{Prosecute}, \mathrm{sid}, \mathrm{EK}, \mathrm{VK}, \pi, C_z^{(t)}, C_x, C_y)$ 作为输入触发仲裁合约 $\mathcal{G}(\mathrm{ConP} - \mathrm{Arbitrate})$。

7) 设置 state := PROSECUTED。

Arbitrate

当从 $\mathcal{G}(\mathrm{ConP} - \mathrm{Arbitrate})$ 收到 $(\mathrm{Arbitrate}, \mathrm{sid}, \mathrm{Arb}_j^t)$，执行以下步骤。

1) 假设 state:=PROSECUTED。

2) 假设 $\upsilon \leqslant \upsilon_e$。

3) 如果 $\mathrm{Arb}_j^t = 1$，设置 $\mathrm{ledger}[D_i] := \mathrm{ledger}[D_i] + \$\mathrm{reward}$；设置 $\mathrm{ledger}[W_j] := \mathrm{ledger}[W_j] + (\$d_i, D_i) + (\$d_j, W_j)$；设置 $\mathrm{Jud}_{i,j}^t := \mathrm{positive}$，$\mathrm{Jud}_{j,i}^t := \mathrm{negative}$。

4) 否则，设置 $\mathrm{ledger}[D_i] := \mathrm{ledger}[D_i] + \$\mathrm{reward} + (\$d_i, D_i) + (\$d_j, W_j)$；设置 $\mathrm{Jud}_{i,j}^t := \mathrm{negative}$，$\mathrm{Jud}_{j,i}^t := \mathrm{positive}$；

存储 $\mathrm{Jud}_{i,j}^t$ 和 $\mathrm{Jud}_{j,i}^t$；设置 state:=ABORTED。

Timer

如果$\upsilon > \upsilon_e$且state!=ABORTED，执行以下步骤。

1) 如果state=CREATED，设置ledger[D_i]:= ledger[D_i]+$reward+ ($d_i$,$D_i$)；设置$\text{Jud}_{i,j}^t$:= negative，$\text{Jud}_{j,i}^t$:= positive。

2) 如果state = INTENT||COMPUTED||PROVED；设置ledger[D_i]:= ledger[D_i]+$reward+($d_i$,$D_i$)+($d_j$,$W_j$)；设置$\text{Jud}_{i,j}^t$:= negative，$\text{Jud}_{j,i}^t$:= positive。

3) 如果state=PROSECUTED，设置ledger[D_i]:= ledger[D_i]+$reward；设置ledger[$W_j$]:= ledger[$W_j$]+($d_i$,$D_i$)+ ($d_j$,$W_j$)；设置$\text{Jud}_{i,j}^t$:= positive，$\text{Jud}_{j,i}^t$:=negative。

4) 存储$\text{Jud}_{i,j}^t$和$\text{Jud}_{j,i}^t$。

5) 设置state:= ABORTED。

7.1.2 仲裁合约

仅当W_j对D_i的验证结果不满时，仲裁合约才被触发。仲裁合约程序ConP – Arbitrate执行如下。

Arbitrate

当从$\mathcal{G}$(ConP – Delegate)收到消息(Prosecute, sid,EK,VK,$\pi, C_z^{(t)}, C_x, C_y$)，执行以下步骤。

1) 运行$\text{Ver}_1^t = \text{Verify}_1(\text{VK}_z, C_z^{(t)})$，$z \in [l]$。

2) 运行$\text{Ver}_2^t = \text{Verify}_2(\text{VK}, \mathcal{C}, \pi)$。

3) 设置$\text{Ver}_j^t = \text{Ver}_1^t \oplus \text{Ver}_2^t$。

4) 将消息(Arbitrate,sid,Arb_j^t)返回至$\mathcal{G}$(ConP – Delegate)。

7.2 协议描述

本文设计的理性委托计算协议包含协商阶段和执行合约阶段，具体描述如下。

7.2.1 协商阶段

此阶段，委托方与计算方根据信誉挑选对方作为合作者，具体流程如下。

1) 委托方D_i将表示任务$F(\cdot)$的关系$\mathcal{R}$及信誉阈值要求$R_{i,\text{th}}^{t-1}$广播至所有计算方$W_j \in \mathcal{W}_{\text{init}}$。

2) 计算方W_j根据信誉模型评估D_i的信誉$R_{j,i}^{t-1,\text{final}}$。其中，参与者的单次交互评价信息可从区块链上获取。若$R_{j,i}^{t-1,\text{final}} \geqslant R_{j,\text{th}}^{t-1}$，则计算方$W_j$同意执行此任务，返回响应消息。将$W_j$加入集合$\mathcal{W}_{\text{res}}$。

3) 委托方D_i根据信誉模型为计算方$W_j \in \mathcal{W}_{\text{res}}$评估信誉$R_{i,j}^{t-1,\text{final}}$。同样，参与者的单次交互评价信息可从区块链上获取。若$R_{i,j}^{t-1,\text{final}} \geqslant R_{i,\text{th}}^{t-1}$，将$W_j$加入集合$\mathcal{W}_{\text{rep}}$。向$\mathcal{W}_{\text{rep}}$中信誉最高的计算方$W_j$发送响应消息。

7.2.2 执行合约阶段

1) 当D_i与W_j均完成押金提交后，D_i私下将输入x以及对应摘要C_x、公开评估密钥EK和公开验证密钥VK发送给W_j。

2) W_j在本地执行任务计算，将结果承诺与中间参数摘要发送至委托合约后，私下将计算结果y和中间参数摘要发送给D_i。

3) 待D_i收到结果后，同样需要对结果进行承诺，并将此承诺分别发送至委托合约和W_j。

4) W_j在本地生成结果证明π，并对其进行承诺得到H_π，将证明承诺H_π发送至委托合约，私下将结果证明π发送给D_i。

5) D_i根据收到的中间参数摘要和结果证明在本地进行摘要验证和证明验证，然后将验证结果发送至委托合约，如果验证结果表明W_j成功完成任务，则进行收益分配和交互评估，合约终止。

6) 倘若W_j未收到预期结果，则将发起诉求请求消息以触发仲裁合约。仲裁合约给出仲裁结果后，根据双方行为分配收益和交互评估，合约终止。

此外，需根据协议形式化相应的用户端程序UserP-FRDC，具体内容如下。

Init

设置$\mathcal{W}_{\text{init}} := \{W_1, \cdots, W_N\}$，$\mathcal{W}_{\text{res}} := \{\}$，$\mathcal{W}_{\text{rep}} := \{\}$。委托方$D_i$执行以下步骤。

Broadcast

当从$\mathcal{Z}$收到(Broadcast,sid,$\mathcal{R}, R_{i,\text{th}}^{t-1}, D_i$)时，将消息(Broadcast,sid,$\mathcal{R}, R_{i,\text{th}}^{t-1}$)广播至所有$W_j \in \mathcal{W}_{\text{init}}$。

Response

当从W_j收到(Response,sid)时，执行以下步骤。

1) 设置$\mathcal{W}_{\text{res}} := \mathcal{W}_{\text{res}} \cup W_j$。

2) 从区块链获取所有关于W_j的交互评估，根据信誉模型计算W_j的信誉，即$R_{i,j}^{t-1,\text{final}}$。

3) 若$R_{i,j}^{t-1,\text{final}} \geqslant R_{i,\text{th}}^{t-1}$，设置$\mathcal{W}_{\text{rep}} := \mathcal{W}_{\text{rep}} \cup W_j$。

4) 将(Response,sid)分别发送给$\mathcal{W}_{\text{rep}}$中信誉最高的W_j。

Create

当从$\mathcal{Z}$收到消息(Create,sid,$x, \upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4, \upsilon_e$,

$\$w_{ij},\$d_i,D_i)$ 时，执行以下步骤。

1) 运行 $(\tau)\leftarrow \mathrm{KeyGen}_1(1^\lambda)$。

2) 计算 $(\mathrm{EK},\mathrm{VK})\leftarrow \mathrm{KeyGen}_2(\tau,\mathcal{R})$。

3)计算 $C_x=\mathrm{Digest}(\mathrm{EK}_x,x,o_x)$，$H_{\mathrm{EK}}=H(\mathrm{EK})$，$H_{\mathrm{VK}}=H(\mathrm{VK})$，其中 $H(\cdot)$ 为哈希函数。

4) 将消息 $(\mathrm{Create},\mathrm{sid},H_{\mathrm{EK}},H_{\mathrm{VK}},C_x,\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_e,\$w_{ij},\$d_i)$ 发送给 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

5) 将 $(\mathrm{Create},\mathrm{sid},H_{\mathrm{EK}},H_{\mathrm{VK}},x,C_x)$ 发送给 W_j。

Compute

当从 W_j 收到 $(\mathrm{Compute},\mathrm{sid},y,\mathcal{C}\setminus(C_y\cup C_x))$ 时，执行以下步骤。

1) 计算 $C_y=\mathrm{Digest}(\mathrm{EK}_y,y,o_y)$。

2) 将消息 $(\mathrm{Compute},\mathrm{sid},C_y)$ 发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

Verify

当从 W_j 收到 $(\mathrm{Prove},\mathrm{sid},\pi)$ 时，执行以下步骤。

1) 假设之前已经收到过消息 $(\mathrm{Compute},\mathrm{sid},y,\mathcal{C}\setminus(C_y\cup C_x))$。

2) 运行 $\mathrm{Ver}_{i,1}^t=\mathrm{Verify}_1(\mathrm{VK}_z,C_z^{(t)})$。

3) 运行 $\mathrm{Ver}_{i,2}^t=\mathrm{Verify}_2(\mathrm{VK},\mathcal{C},\pi)$。

4) 设置 $\mathrm{Ver}_{i,j}^t=\mathrm{Ver}_{i,1}^t\oplus\mathrm{Ver}_{i,2}^t$。

5) 将消息 $(\mathrm{Verify},\mathrm{sid},\mathrm{Ver}_{i,j}^t)$ 发给 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

计算方 W_j 执行以下步骤。

Response

当从 D_i 收到 $(\mathrm{Broadcast},\mathrm{sid},\mathcal{R},R_{i,\mathrm{th}}^{t-1})$ 时，执行以下步骤。

1) 从区块链获取所有关于 D_i 的交互评估，根据信誉模型计算 D_i 的信誉，即 $R_{j,i}^{t-1,\mathrm{final}}$。

2) 若 $R_{j,i}^{t-1,\mathrm{final}}\geqslant R_{j,\mathrm{th}}^{t-1}$，将 $(\mathrm{Response},\mathrm{sid})$ 发至 D_i。

Intent

当从 $\mathcal{Z}$ 收到消息 $(\mathrm{Intent},\mathrm{sid},W_j)$ 时，将消息 $(\mathrm{Intent},\mathrm{sid},\$d_j)$ 发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

Compute

当从 $\mathcal{Z}$ 收到 $(\mathrm{Compute},\mathrm{sid},W_j)$ 时，执行以下步骤。

1) 假设之前已经收到过 $(\mathrm{Intent},\mathrm{sid},\$d_j)$。

2) 计算 $C_z^{(t)}=\mathrm{Digest}(\mathrm{EK}_z,\chi_z^{(t)},o_z^{(t)})$，$H_y=H(y)$。

3) 将消息 $(\mathrm{Compute},\mathrm{sid},H_y,\mathcal{C}\setminus(C_y\cup C_x))$ 发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

4) 将 $(\mathrm{Compute},\mathrm{sid},y,\mathcal{C}\setminus(C_y\cup C_x))$ 发至 D_i。

Prove

当从 $\mathcal{Z}$ 收到 $(\mathrm{Prove},\mathrm{sid},W_j)$ 时，执行以下步骤。

1) 运行算法执行 $\pi\leftarrow\mathrm{Prove}(\mathrm{EK},\chi,\mathcal{O})$，计算 $H_\pi=H(\pi)$。

2) 将消息 $(\mathrm{Prove},\mathrm{sid},H_\pi)$ 发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$

3) 将 $(\mathrm{Prove},\mathrm{sid},\pi)$ 发送至 D_i。

Prosecute

若 D_i 未在 υ_4 前将消息 $(\mathrm{Verify},\mathrm{sid},1)$ 发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$，则将 $(\mathrm{Prosecute},\mathrm{sid},\mathrm{EK},x,y,\pi,\$h_j)$ 发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

8 方案分析

8.1 安全性分析

定理 1 现实世界中的协议安全实现理想函数 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$。

证明 本文采用文献[21]中的证明框架完成协议的 UC 安全性证明。对于任意现实敌手 $\mathcal{A}$，需构造一个理想世界模拟器 $\mathcal{S}$，使任意多项式时间环境 $\mathcal{Z}$ 无法以一个不可忽略的概率区分现实世界与理想世界的执行。下文将构造模拟器 SimP 的用户端部分，模拟器封装器 $\mathcal{S}(\cdot)$ 可将模拟器转换成理想敌手 $\mathcal{S}(\mathrm{SimP})$，简写为 $\mathcal{S}$。

仿真委托方 D_i 如下。

1) 当环境 $\mathcal{Z}$ 将消息 $(\mathrm{Create},\mathrm{sid},x,\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_e,\$w_{ij},\$d_i,D_i)$ 发送至 D_i 时，Sim$P$ 从理想函数 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$ 收到 $(\mathrm{Create},\mathrm{sid},|x|,\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_e,\$w_{ij},\$d_i,D_i)$。SimP 运行 $(\tau)\leftarrow\mathrm{KeyGen}_1(1^\lambda)$ 和 $(\mathrm{EK},\mathrm{VK})\leftarrow\mathrm{KeyGen}_2(\tau,\mathcal{R})$。SimP 对向量 **0** 计算摘要，即 $C_{x^0}=\mathrm{Digest}(\mathrm{EK}_{x^0},\mathbf{0},O_{x^0})$。SimP 分别对 EK 和 VK 进行哈希承诺，即 $H_{\mathrm{EK}}=H(\mathrm{EK})$，$H_{\mathrm{VK}}=H(\mathrm{VK})$。SimP 将 $(\mathrm{Create},\mathrm{sid},C_{x^0},H_{\mathrm{EK}},H_{\mathrm{VK}},\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_e,\$w_{ij},\$d_i)$ 发送至内部仿真的合约 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

2) 一旦从 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$ 处收到消息 $(\mathrm{Compute},\mathrm{sid},W_j)$，SimP 对向量 **0** 计算摘要，即 $C_{y^0}=\mathrm{Digest}(\mathrm{EK}_{y^0},\mathbf{0},O_{y^0})$。SimP 将消息 (Compute,

$\mathrm{sid}, C_{y^0})$ 发送至仿真的 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

3) 当环境 $\mathcal{Z}$ 将消息 $(\mathrm{Verify},\mathrm{sid}, D_i)$ 发送至 D_i 时，SimP 从 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$ 收到 $(\mathrm{Verify},\mathrm{sid}, D_i)$。SimP 将消息 $(\mathrm{Verify},\mathrm{sid},1)$ 发送至仿真的 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

仿真计算方 W_j 如下。

1) 当环境 $\mathcal{Z}$ 将消息 $(\mathrm{Intent},\mathrm{sid},\$d_j, W_j)$ 发送至 W_j 时，SimP 通过 $\mathcal{S}$ 路由从 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$ 处收到 $(\mathrm{Intent},\mathrm{sid},\$d_j, W_j)$，并将此消息转发给内部仿真的 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

2) 当环境 $\mathcal{Z}$ 将消息 $(\mathrm{Compute},\mathrm{sid}, W_j)$ 发送至 W_j 时，SimP 从 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$ 处收到消息 $(\mathrm{Compute},\mathrm{sid}, W_j)$。SimP 分别对向量 $\mathbf{0}$ 进行 $l-2$ 次摘要计算，即每次根据不同的摘要公钥和随机数计算 $C_{z^0}^{(t)} = \mathrm{Digest}(\mathrm{EK}_{z^0}, \mathbf{0}, o_{z^0}^{(t)})$。SimP 对向量 $\mathbf{0}$ 进行哈希承诺，即 $H_{y^0} = H(\mathbf{0})$。SimP 将 $(\mathrm{Compute},\mathrm{sid}, H_{y^0}, \mathcal{C}\setminus(C_{y^0}\cup C_{x^0}))$ 发送至内部 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

3) 一旦从 $\mathcal{F}(\mathrm{Ideal}-\mathrm{FRDC})$ 处收到消息 $(\mathrm{Prove},\mathrm{sid}, W_j)$，SimP 运行 $\pi_0 = \mathrm{Prove}(\mathrm{EK}, \chi_0, \mathcal{O}_0)$，并对证明进行哈希承诺 $H_{\pi_0} = H(\pi_0)$。SimP 将消息 $(\mathrm{Prove},\mathrm{sid}, H_{\pi_0})$ 发送至内部 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。

现实世界与理想世界的不可区分性。

现实世界（Real）。首先定义仅存在虚拟敌手的现实世界，此敌手只转发来自环境 $\mathcal{Z}$ 的消息，或将消息传递给 $\mathcal{Z}$。

混合 1（Hybrid 1）。Hybrid 1 与 Real 相同除了以下区别：敌手（或被称为模拟器）执行 SCP 方案中仿真的密钥产生阶段，即 $(\tau)\leftarrow \mathrm{KeyGen}_1(1^\lambda)$ 和 $(\mathrm{EK},\mathrm{VK})\leftarrow \mathrm{KeyGen}_2(\tau,\mathcal{R})$。当诚实的 W_j 产生证明时，模拟器通过调用 $\mathrm{Prove}(\mathrm{EK},\cdot,\cdot)$ 算法得到仿真证明，将仿真证明发送至环境 $\mathcal{Z}$，此过程不泄露任何秘密信息。当诚实的 D_i 对输入数据 x 与计算结果 y 进行摘要计算，诚实的 W_j 对中间数据进行摘要计算时，模拟器通过调用 $\mathrm{Digest}(\mathrm{EK},\cdot,\cdot)$ 算法得到仿真摘要，将仿真摘要发送至环境 $\mathcal{Z}$。

事件 1（Fact 1）。因本文采用的 SCP 方案具有完美零知识性和绑定承诺，故不存在多项式时间环境 $\mathcal{Z}$ 能以不可忽略的概率区分 REAL 与 Hybrid 1。

混合 2（Hybrid 2）。模拟器仿真合约函数 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$。因为发送至 $\mathcal{G}(\mathrm{ConP}-\mathrm{Delegate})$ 的所有消息都是公开的，所以仿真合约函数是简单的。故对于环境 $\mathcal{Z}$ 而言，Hybrid 2 与 Hybrid 1 具有相同分布。

混合 3（Hybrid 3）。Hybrid 3 与 Hybrid 2 相同除了以下区别：当诚实的 D_i 对评估密钥 EK 和验证密钥 VK 进行哈希承诺，诚实的 W_j 对计算结果 y 进行哈希承诺时，模拟器通过调用 $H(\cdot)$ 算法得到仿真哈希承诺，将仿真哈希承诺发送至环境 $\mathcal{Z}$。

事件 2（Fact 2）。如果哈希函数是安全的，则不存在多项式时间环境 $\mathcal{Z}$ 能以不可忽略的概率区分 Hybrid 3 与 Hybrid 2。

证毕。

定理 2　如果 $w_{ij} < d_i$，{诚实,支付} 为公平理性委托计算博弈模型 G_t 中的唯一子博弈精炼纳什均衡。

证明　本文采用逆向归纳法求解博弈模型 G_t 的子博弈精炼纳什均衡。

仅当委托方 D_i 选择拒绝支付时，委托方 W_j 才会进行第二轮策略选择。当 W_j 在第一轮选择诚实策略，D_i 选择拒绝策略时，W_j 选择起诉或沉默策略分别可获得收益 $d_i - c_j - h_j$ 和 $-c_j$（$d_i - c_j - h_j > -c_j$），故理性的 W_j 会选择起诉对应终端节点 v_2），此时 D_i 相应的收益为 $f_i - d_i - c_i$。当 W_j 在第一轮选择恶意策略，D_i 选择拒绝策略时，W_j 选择起诉或沉默策略分别可获得收益 $-h_j - d_j$ 和 $-d_j$（$-h_j - d_j < -d_j$），故理性的 W_j 会选择沉默，此时 D_i 相应的收益为 $d_j - c_i$。

D_i 需在 W_j 做出第一轮决策后选择自己的行动。当 W_j 选择诚实策略时，D_i 选择支付或拒绝策略分别可获得收益 $f_i - w_{ij} - c_i$ 和 $f_i - d_i - c_i$（$f_i - w_{ij} - c_i > f_i - d_i - c_i$），故理性的 D_i 会选择支付，此时 W_j 相应的收益为 $w_{ij} - c_j$。当 W_j 选择恶意策略时，D_i 选择支付或拒绝策略分别可获得收益 $-w_{ij} - c_i$ 和 $d_j - c_i$（$-w_{ij} - c_i < d_j - c_i$），故理性的 D_i 会选择拒绝，此时 W_j 相应的收益为 $-d_j$。

W_j 在已知自己选择诚实策略下的最终收益为 $f_i - w_{ij} - c_i$，而选择恶意策略的最终收益为 $-d_j$。

由于 $w_{ij}-c_j>-d_j$，理性的 W_j 将会选择诚实策略。基于上述分析，理性的 D_i 将会选择支付策略。故该博弈模型下的唯一子博弈纳什均衡为 {诚实,支付}，对应博弈模型中的终端节点 v_1。其结果为计算方 W_j 将诚实执行计算任务，委托方 D_i 将支付委托费用，意味着该场景下的理性参与者将诚实执行委托合约，不触发仲裁合约。

证毕。

8.2 方案对比

本节将从公开可验证性、经济公平性、信誉公平性、博弈模型以及 UC 安全性方面与现有的委托计算方案进行对比，结果如表 3 所示。其中，√表示该方案满足此性质，×表示不满足此性质，—表示不涉及此性质。

表 3 本文方案与其他方案性能对比

方案	公开可验证性	经济公平性	信誉公平性	博弈模型	UC 安全性
文献[8]方案	—	×	—	√	—
文献[9]方案	—	√	—	√	—
文献[10]方案	—	×	—	√	—
文献[13]方案	—	√	√	√	—
文献[17]方案	—	√	×	√	—
文献[22]方案	√	√	—	×	—
本文方案	√	√	√	√	√

文献[8,10]方案均考虑诚实的委托方将任务委托给 2 个自利计算方，通过检查计算方返回的结果是否一致来判断两方是否存在不诚实行为。2 种方案都构建了相应的博弈模型，但不支持经济公平性，也不涉及公开可验证性、信誉公平性以及 UC 安全性的讨论。而文献[9]方案改进了文献[8]方案，提出了公平支付协议。

文献[13,17]方案考虑委托方与计算方均为理性的情况，分别构建对应场景下的博弈模型且支持双方的公平支付。文献[13]方案考虑用信誉激励委托方与计算方，该方案保证了信誉公平性。但是 2 种方案均不涉及公开可验证性与 UC 安全性的讨论。

由于本文考虑的是基于证明的理性委托计算，因此仅与文献[22]方案进行比较。文献[22]方案基于文献[6]中的 Pinocchio 协议实现了对计算方返回结果的公开验证，此外，利用智能合约实现了双方的公平支付。但是该方案并没有构建博弈模型，也没有考虑协议的 UC 安全性和信誉公平性。

本文考虑委托方与计算方均为理性且具有信誉激励的情况，利用博弈论构建具有完美信息的两方动态博弈模型。本文采用文献[7]中的 Geppetto 协议实现计算结果的公开可验证，利用智能合约实现了经济公平性和信誉公平性，并构建相应的可组合安全模型，证明协议的 UC 安全性。

9 结束语

本文利用密码学的区块链模型探究在 UC 框架下的公平理性委托计算方案。首先，结合直接信誉和间接信誉构建双向激励信誉模型，委托方与计算方均可根据自己的信誉要求选择合适的合作方。其次，将委托方与计算方均视为理性参与者，基于博弈论构建理性委托计算场景下具有完美信息的两方动态博弈模型。再次，通过分析该场景下安全需求、理性需求与敌手模型，构建可组合安全模型，即公平理性委托计算理想函数。最后，通过结合简洁承诺证明与智能合约技术，设计可安全实现理想函数的公平理性委托计算协议。该协议具有可公开验证性、经济公平性、信誉公平性和 UC 安全性，且当参与者均为理性时，双方均选择诚实执行协议形成唯一子博弈精炼纳什均衡。

参考文献：

[1] GOLDWASSER S, KALAI Y T, ROTHBLUM G N. Delegating computation: interactive proofs for muggles[C]//Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. New York: ACM Press, 2008: 113-122.

[2] GOLDWASSER S, MICALI S, RACKOFF C. The knowledge complexity of interactive proof systems[J]. SIAM Journal on Computing, 1989, 18(1): 186-208.

[3] ARORA S, SAFRA S. Probabilistic checking of proofs: a new characterization of NP[J]. Journal of the ACM, 1998, 45(1): 70-122.

[4] BITANSKY N, CANETTI R, CHIESA A, et al. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again[C]//Innovations in Theoretical Computer Science Conference. New York: ACM Press, 2012: 326-349.

[5] GENNARO R, GENTRY C, PARNO B, et al. Quadratic span programs and succinct NIZKs without PCPs[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2013: 626-645.

[6] PARNO B, HOWELL J, GENTRY C, et al. Pinocchio: nearly practical verifiable computation[C]//2013 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2013: 238-252.

[7] COSTELLO C, FOURNET C, HOWELL J, et al. Geppetto: versatile verifiable computation[C]//2015 IEEE Symposium on Security and

Privacy. Piscataway: IEEE Press, 2015: 253-270.

[8] BELENKIY M, CHASE M, ERWAY C C, et al. Incentivizing outsourced computation[C]//Proceedings of the 3rd International Workshop on Economics of Networked Systems. New York: ACM Press, 2008: 85-90.

[9] KUPCU A. Incentivized outsourced computation resistant to malicious contractors[J]. IEEE Transactions on Dependable and Secure Computing, 2017, 14(6): 633-649.

[10] DONG C Y, WANG Y L, ALDWEESH A, et al. Betrayal, distrust, and rationality: smart counter-collusion contracts for verifiable cloud computing[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2017: 211-227.

[11] TIAN Y L, GUO J, WU Y L, et al. Towards attack and defense views of rational delegation of computation[J]. IEEE Access, 2019, 7: 44037-44049.

[12] ZHANG Y, VAN D S M. Reputation-based incentive protocols in crowdsourcing applications[C]//2012 Proceedings IEEE INFOCOM. Piscataway: IEEE Press, 2012: 2140-2148.

[13] MA X D, MA J F, LI H, et al. RTRC: a reputation-based incentive game model for trustworthy crowdsourcing service[J]. China Communications, 2016, 13(12): 199-215.

[14] JIANG X X, TIAN Y L. Rational delegation of computation based on reputation and contract theory in the UC framework[C]// International Conference on Security and Privacy in Digital Economy. Berlin: Springer, 2020: 322-335.

[15] CANETTI R. Universally composable security: a new paradigm for cryptographic protocols[C]//Proceedings 42nd IEEE Symposium on Foundations of Computer Science. Piscataway: IEEE Press, 2001: 136-145.

[16] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system[R]. (2008-10-31) [2021-01-04].

[17] 李沓, 田有亮, 向康, 等. 委托计算下基于区块链的公平支付方案[J]. 通信学报, 2020, 41(3): 80-90.
LI T, TIAN Y L, XIANG K, et al. Block-based fair payment scheme under delegation computation[J]. Journal on Communications, 2020, 41(3): 80-90.

[18] 尹鑫, 田有亮, 王海龙. 公平理性委托计算协议[J]. 软件学报, 2018, 29(7): 1953-1962.
YIN X, TIAN Y L, WANG H L. Fair and rational delegation computation protocol[J]. Journal of Software, 2018, 29(7): 1953-1962.

[19] ZHANG Y H, DENG R H, LIU X M, et al. Blockchain based efficient and robust fair payment for outsourcing services in cloud computing[J]. Information Sciences, 2018, 462: 262-277.

[20] KOSBA A, MILLER A, SHI E, et al. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts[C]//2016 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2016: 839-858.

[21] JUELS A, KOSBA A, SHI E. The ring of gyges: investigating the future of criminal smart contracts[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 283-295.

[22] DORSALA M R, SASTRY V N, CHAPRAM S. Fair payments for verifiable cloud services using smart contracts[J]. Computers & Security, 2020, 90: 101712.

[23] 张维迎. 博弈论与信息经济学[M]. 上海: 格致出版社, 2012.
ZHANG W Y. Game theory and information economics[M]. Shanghai: Gezhi Press, 2012.

[24] DELMOLINO K, ARNETT M, KOSBA A, et al. Step by step towards creating a safe smart contract: lessons and insights from a cryptocurrency lab[C]//2016 International Conference on Financial Cryptoraphy and Data Security. Berlin: Springer, 2016: 79-94.

[25] HUANG K L, KANHERE S S, HU W. On the need for a reputation system in mobile phone based sensing[J]. Ad Hoc Networks, 2014, 12: 130-149.

[26] KENNEY J F, KEEPING E S. Mathematics of statistics-part one[M]. New York: D. van Nostrand, 1954.

[27] XIONG L, LIU L. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(7): 843-857.

[28] KANG J W, XIONG Z H, NIYATO D, et al. Incentive mechanism for reliable federated learning: a joint optimization approach to combining reputation and contract theory[J]. IEEE Internet of Things Journal, 2019, 6(6): 10700-10714.

[作者简介]

田有亮（1982− ），男，贵州盘县人，博士，贵州大学教授，主要研究方向为博弈论、密码学与安全协议。

蒋小霞（1997− ），女，贵州遵义人，贵州大学硕士生，主要研究方向为通用可组合安全、理性委托计算。