软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(3):1491−1511 [doi: 10.13328/j.cnki.jos.006716]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于智能合约的工业互联网数据公开审计方案*

李 涛[1,2,3], 杨安家[1,2,3], 翁 健[1,2,3], 郭梓繁[1,2,3]

[1](暨南大学 网络空间安全学院, 广东 广州 510632)

[2](网络安全检测与防护技术国家地方联合工程研究中心(暨南大学), 广东 广州 510632)

[3](广东省数据安全与隐私保护重点实验室(暨南大学), 广东 广州 510632)

通信作者: 杨安家, E-mail: ayang3-c@my.cityu.edu.hk

摘 要: 随着工业互联网产生的数据量日益增加, 越来越多的企业选择将工业互联网数据外包存储在云服务器上以节省存储开销. 为了防止外包存储的数据被篡改或删除, 企业需要定期对其进行审计. 提出了一种基于智能合约的工业互联网数据公开审计方案. 该方案基于博弈论的思想, 设计了一系列智能合约, 以高效地抵抗参与者恶意行为. 与现有抗合谋的公开审计方案相比, 该方案不依赖于复杂的密码学工具实现对参与者恶意行为的抵抗, 使得其更为高效, 进而能够更好地应用于海量且频繁更新的工业互联网数据场景中. 特别地, 所设计的博弈合约作为一种独立的工具, 能够与现有的公开审计方案有效结合, 在不降低其审计效率的同时, 增加方案的安全性. 在本地环境和以太坊公有测试链 Ropsten 上对博弈合约以及整体方案进行了一系列的测试, 结果表明, 所设计的合约运行花费低且对运行环境适应性强, 对原有完整性审计方案的效率影响小; 同时, 与其他抗审计者恶意行为的完整性方案相比, 该方案更为高效.

关键词: 工业互联网; 云存储; 公开审计; 智能合约; 博弈论

中图法分类号: TP309

中文引用格式: 李涛, 杨安家, 翁健, 郭梓繁. 基于智能合约的工业互联网数据公开审计方案. 软件学报, 2023, 34(3): 1491−1511. http://www.jos.org.cn/1000-9825/6716.htm

英文引用格式: Li T, Yang AJ, Weng J, Guo ZF. Public Auditing Scheme for Industrial Internet Data Based on Smart Contracts. Ruan Jian Xue Bao/Journal of Software, 2023, 34(3): 1491−1511 (in Chinese). http://www.jos.org.cn/1000-9825/6716.htm

Public Auditing Scheme for Industrial Internet Data Based on Smart Contracts

LI Tao[1,2,3], YANG An-Jia[1,2,3], WENG Jian[1,2,3], GUO Zi-Fan[1,2,3]

[1](College of Cyber Security, Jinan University, Guangzhou 510632, China)

[2](National Joint Engineering Research Center of Network Security Detection and Protection Technology (Jinan University), Guangzhou 510632, China)

[3](Guangdong Provincial Key Laboratory of Data Security and Privacy Protection (Jinan University), Guangzhou, 510632, China)

Abstract: As the amount of data generated by the Industrial Internet grows, more and more companies are choosing to outsource the storage for their Industrial Internet data to cloud servers to save storage costs. To prevent the outsourced data from being tampered or deleted, companies need to audit the data integrity regularly. This study proposes a public auditing scheme for Industrial Internet data based on smart contracts. Particularly, a series of game-theory based smart contracts are designed which can efficiently mitigate malicious participators including the third-party auditor and the cloud server. Compared to existing collusion-resistant public auditing schemes, the proposed scheme does not rely on complex cryptographic tools to achieve resistance to participant malicious behavior, and thus is more efficient and suitable to Industrial Internet applications where huge amount of data need to be frequently updated. Specifically, the

* 基金项目: 广东省重点领域研发计划(2020B0101360001); 国家重点研发计划(2021ZD0112802, 2020YFB1005600, 2017YFB0802200, 2018YFB100370); 国家自然科学基金(62072215, U1736203, 61825203)

收稿时间: 2022-01-06; 修改时间: 2022-02-21; 采用时间: 2022-05-06; jos 在线出版时间: 2022-07-22

game-based contract designed in this study as an individual solution, can be effectively combined with existing public auditing schemes to turn out a public auditing scheme with better security without losing efficiency. Finally, a series of tests are conducted on the proposed contract in the local environment and Ropsten, the common test chain for Ethereum. The results show that the designed contract is cheap to run and adaptable to the operating environment, has little impact on the efficiency of the original integrity audit solution, and is more efficient than other integrity schemes that resist the malicious behavior of auditors.

Key words: industrial Internet; cloud storage; public auditing; smart contract; game theory

随着信息技术在制造业的应用, 工业互联网成为备受关注的热点[1]. 工业互联网每日都会产生海量数据, 这些数据包含的价值能用于解决工业制造流程问题, 优化工业制造结构, 从而推动工业发展[2]. 为了充分地挖掘工业互联网数据的价值, 企业需要先将其进行存储. 然而, 工业互联网数据具有时效性和弱事务性的特征, 传统的数据存储方法更倾向于存储具有延期性和强事务性特征的数据. 因此, 传统的数据存储方法并不适用于存储工业互联网数据. 如何存储工业互联网数据, 已经成为工业互联网发展路上亟需解决的难题[3].

云存储技术是指数据提供方雇佣云服务器为其进行数据存储服务. 云存储相比传统本地数据库存储更便捷, 节省了企业本地购入存储设备以及后期维护的大额开销. 同时, 云存储还具有强可塑性, 它能为不同的客户制定不同需求的服务, 非常契合工业互联网数据存储的要求. 然而, 将工业互联网数据外包给云存储服务器后[4−6], 企业失去了对数据的控制. 如何确保云存储服务器不会删除或修改数据, 即确保外包数据的完整性, 成为新的技术挑战.

为了检验在云存储服务器数据的完整性, 在 2007 年, Juels 等人[7]和 Ateniese 等人[8]分别提出了 PoR (proofs of retrievability)和 PDP (provable data possession)的概念. 其基本思想为: 对存储在云服务器上的外包数据进行抽样审计, 在一定轮数的检查与验证之后, 能够以极大的概率确定数据的完整性. 其中, Ateniese 等人[8]还首次考虑了公开审计的模式, 即除了数据拥有者之外, 其他任何第三方(third party auditor, TPA)也可以对外包存储数据的完整性进行检验, 这使数据拥有者不再需要承担沉重的审计工作. 而工业互联网中的企业每天都需要进行工业制造流程的管理, 相比于私有审计的方案, 使用可公开检验的审计方案可以节省它们的精力.

由于其具有较强的灵活性, 公开审计方案近几年受到了众多学者的关注, 出现了许多相关研究[9−24]. 当前, 大部分可公开审计方案都在半可信审计者的情况下执行协议, 即审计者会诚实地执行审计的流程, 但可能尝试窥探外包数据的隐私. 然而, 在实际场景中, 审计者以利益作为驱动力, 因此其更有可能为了获取更高利益违背审计流程. 例如, 在工业互联网的场景中, 理性的审计者面对频繁更新且海量的工业互联网数据需要多次执行审计工作, 其可以进行恶意不执行或延迟审计以节约审计成本, 甚至它还可以与云存储服务器合谋欺骗用户. 这些攻击给工业互联网企业外包存储数据的安全性造成巨大威胁. 因此, 在设计方案时需要进一步考虑恶意审计者的情况. 虽然一部分学者进行了相关的研究, 但是为了抵抗恶意审计者, 他们使用了具有更多性质的高级密码学工具, 导致方案效率不够高, 不能很好地应用于工业互联网数据场景中. 例如, 文献[25]中提出了使用无证书签名的审计方案, 以达到抵御审计者恶意行为的目的. 然而, 该方案的效率相比使用传统简单密码学工具的完整性审计方案要低. 如果将其使用在海量且频繁更新的工业互联网数据场景中, 会加大审计者的工作量, 从而增加企业的开销. 因此, 工业互联网的数据完整性审计方案需要满足在能够抵御审计者恶意行为的同时, 还不会增加企业过多的额外开销.

文献[26]将博弈论技术与区块链技术结合, 提出了一个不引入复杂密码学工具也能解决外包云计算场景中理性的云服务器恶意行为以及合谋问题的方案. 因此, 基于博弈论的云计算方案在保证其安全性的同时, 还具有其他基于纯密码学方案所不能实现的高效性. 博弈论提供的安全与高效特性与工业互联网数据场景的数据海量且频繁更新特性契合, 其为实现安全高效的工业互联网数据完整性审计方案提供了新的解题思路.

然而, 该思路在应用于工业互联网外包数据公开审计场景前, 仍需要解决以下 3 个技术挑战.

1) 工业互联网外包数据公开审计场景中, 恶意且理性的审计者和云存储服务器会发起的攻击与外包云计算场景的大不相同. 如, 由于工业互联网数据更新频繁的特性, 导致审计者需要短时间内进行

多次审计, 审计者可能会发起的恶意延迟审计攻击或不审计以节省审计成本, 云存储服务器也可能发起的恶意扰乱审计流程攻击以设法获得审计者的押金, 甚至审计者和云存储服务器可以共同发起的恶意合谋攻击欺骗用户. 具体实现的完整性审计方案应能抵御在工业互联网场景下理性的审计参与者所能发起的全部攻击.

2) 文献[26]解决的是在外包云计算场景里两个相同地位的理性云计算服务器发生恶意行为与合谋的问题, 而在工业互联网外包数据公开审计场景中, 审计者和云存储服务器的地位并不相同, 即它们具有不同的获取利益手段, 对应的博弈合约流程也并不相同. 因此, 所实现的完整性审计方案需要分别考虑各方的利益, 从而分析得出它们在每个博弈合约中会做出的各种行为, 以优化博弈合约流程, 实现能促使双方诚实地执行审计流程的博弈合约.

3) 在外包云计算场景中, 云计算服务器最终会返回一个具体的结果给用户, 因此文献[26]所设计的博弈只需要直接对比最后的结果, 而并没有与现有云计算方案进行有机结合. 然而在工业互联网外包数据公开审计场景中, 审计者并不会返回一个具体的结果给用户, 而是返回审计流程记录. 如果用户直接使用智能合约来验证这些记录的正确性, 则会造成巨额的智能合约运行花费. 因此, 应寻找一种方式将智能合约与完整性审计方案有机地结合起来, 并在不影响完整性审计方案效率的前提下, 将智能合约变为审计流程中的一个部分, 以节约其验证工作量, 进一步减少其运行花费.

在此基础上, 本文提出了基于博弈合约的安全高效工业互联网外包数据公开审计方案. 该方案分为博弈合约部分和完整性审计部分. 其中, 博弈合约部分借鉴文献[26]的思想, 从博弈论的角度出发, 引入并设计了 3 个智能合约, 能够抵御理性的恶意攻击者, 包括恶意的审计者与云存储服务器. 特别地, 博弈合约部分在方案中作为一个安全工具, 其作用是为原本不能抵御审计者恶意行为的完整性审计方案提供额外安全特性. 因此, 本方案的完整性审计部分具有高拓展性, 任何一个符合一定形式的公开审计方案都可与我们提出的博弈合约有机结合. 此外, 我们设计的智能合约运行高效, 以保证在不降低原本公开审计方案审计效率的同时增加其安全性. 本文的主要贡献包括 3 个方面.

1) 从博弈论的角度分析工业互联网公开审计方案中审计者和云存储服务器的利益关系, 对恶意审计者和云存储服务器发起攻击与否时的利益得失进行建模, 根据收益模型设计一套基于智能合约的奖惩机制, 该奖惩机制由囚徒合约、合谋合约和背叛合约组成. 它们使得双方都诚实地执行协议时才能获得最大利益, 因此降低了恶意参与者的攻击动机.

2) 基于以上几个合约, 设计出一个能够抵抗恶意审计者和云存储服务器的公开审计方案. 具体地, 详细给出了如何将以上博弈合约与现有的公开审计方案有机结合的方法, 展示出本方案的高拓展性.

3) 在本地环境与以太坊公有测试链上测试了本文设计智能合约的效率, 并得出其具备高效性以及对运行环境适应性强的结论.

本文第 1 节介绍公开完整性验证的相关方法和研究现状. 第 2 节介绍本文所需的储备知识, 包括博弈论模型和智能合约的定义. 第 3 节介绍本文所构建的系统模型, 包括系统角色的介绍、威胁模型的介绍和安全目标的定义. 第 4 节给出我们方案的一个具体例子. 第 5 节对我们的方案进行安全分析, 以证明本文的方案能抵御系统模型中定义的威胁模型. 第 6 节对本文的方案进行性能测试, 以证明其高效性与高适应性. 最后总结全文.

1 相关工作

自 Ateniese 等人[8]提出公开可验证的审计方案以来, 涌现了大批优秀的国内外研究者对该问题进一步地研究[9−24]. 文献[9–15]沿着 Juels 等人[7]提到的动态数据审计的研究思路, 考虑外包数据的动态更新需求, 设计了多种支持动态更新的审计方案. 随着公开审计研究的进行, 其优势逐步显现, 然而也产生了新的安全挑战, 即第三方审计者的诚实问题. 文献[16,17,27]考虑了半可信的第三方审计者, 即第三方审计者会诚实地执行任务但保留执行过程中通过窃听所获得的数据, 提出了不同的具有隐私保护性质的公开审计方案. 随着半可信

模型下的完整性审计方案被逐步完善, 学者们开始解决公开审计在实际应用中所遇到的问题. 例如, 敌手恶意地频繁向存储服务器发起审计请求, 导致用户需要支付的网络费用和审计费用激增, 甚至导致审计任务中断的问题. Yang 等人[9,10]提出了代理审计的概念, 数据拥有者将审计任务代理给指定的第三方审计者, 该方案在数据标签生成方面具有很高的效率. 伴随着研究工作的进一步深入, 产生了大量适用于不同场景的公开审计方案. 文献[18,19]考虑在云存储场景下的多副本外包数据的公开审计方案, 由于云存储服务商往往是采用分布式存储和 RAID (redundant array of independent disks)结合的方式进行存储, 其特殊的存储方式要求公开审计方案能够对外包存储在多个备份云服务器上的数据同时进行完整性验证. 文献[20]提出一种自适应数据持有证明方法以提高审计效率, 该方法能够基于文件属性和用户需求动态地调整文件的审计方案, 适用于对数据安全性有较高要求的场景. He 等人[21]提出一个共享数据的公开审计方案, 该方案支持多个用户对外包数据进行更新.

以上公开审计方案都假设第三方审计者半可信, 但在实际情况中, 审计者有着很强的经济动机做出恶意行为, 例如延迟审计、不审计、甚至与服务器合谋来获取更大的利益. 为此, Xu[22]引入了时间服务器的技术, 用来抵御审计者和云存储服务器的合谋攻击, 但该方案带来时间服务器维护的问题, 同时无法有效抵抗延迟审计攻击. Armknecht 等人在文献[28,29]中也考虑了审计者不可信的问题, 可以防止恶意审计者、恶意用户以及恶意云存储服务提供方任意两者之间的合谋攻击, 但是这些方案无法抵抗延迟审计攻击. Zhang 等人[25]提出了基于无证书的公开审计方案, 解决了延迟审计攻击的问题. 然而, 该方案的验证阶段比传统方案增加一些额外的计算开销, 为了解决该问题, 他们在文献[30]中通过区块链技术来抵御延迟审计攻击, 但是该方案仍然不够高效. 综上所述, 现今仍然缺少能高效地同时防范合谋攻击和延迟审计攻击的外包数据公开审计方案.

2 储备知识

在本节中, 我们介绍本文所使用的博弈与策略概念和智能合约概念.

2.1 博弈与策略

在本文中, 我们参考文献[26]中对博弈与策略的定义, 使用博弈树来表示审计者和数据存储服务器双方的博弈关系. 我们考虑博弈参与方都是理性即以自身利益优先, 同时在博弈树中假设双方无法得知对方做出的选择, 它们完全基于自己的最高收益而做出决定. 具体的博弈定义如下.

定义 1. 本文提到的博弈, 可以使用这样的一个元组表示: $G=(P,A,H,Z,x,\rho,\sigma,u,I)$, 其中,

- P 代表参与者集合.
- A 代表行动集合.
- H 代表非终端节点集合.
- Z 代表终端节点集合.
- $x:H\to 2^A$, 代表行动函数, 给每一个非终端节点分配一组可能的行动.
- $\rho:H\to N$, 代表参与者分配函数, 给每一个非终端节点安排其归属的参与者 i, $i\in N$.
- $\sigma:H\times A\to H\cup Z$, 代表一个继承函数, 将每一个非终端节点以及其对应的行动映射到一个新的非终端节点或者终端节点. 继承函数具有唯一性, 即对于任意的两个非终端节点 $h_1,h_2\in H$ 以及其对应的行动 $a_1, a_2\in A$, 如果$\sigma(h_1,a_1)=\sigma(h_2,a_2)$, 那么 $h_1=h_2$, $a_1=a_2$.
- $u=(u_1,\ldots,u_n)$, 其中, $u_i:Z\to R$, 代表参与者 i 在终端节点集合 Z 中获得的实际收益.
- $I=(I_1,\ldots,I_n)$, 其中, I_i 是参与者 i 与非终端节点的一个等价关系, 即将参与者 i 拥有的非终端节点分为 k_i 份, $I_{i,1},\ldots,I_{i,k_i}$. 每一份集合中的任意两个非终端节点都具有这两个性质:

$$x(h)=x(h'),\ \rho(h)=\rho(h')(h,h'\in I_{i,j}).$$

接下来, 我们将用一个博弈例子来详细介绍定义 1, 如图 1 所示. 在该博弈树中, 我们使用圆形图案表示非终端节点, 方形图案表示终端节点, 并且每个节点都有它们的标签 v_i. 该博弈中存在着两个参与者, 即

$P=\{P_1,P_2\}$, 它们可以采取的行动集合为 $A=\{Y,W,N,o,q,i,j\}$. 非终端节点的集合 $H=(v_0,v_1,v_2,v_3,v_4,v_5,v_6,v_7)$, 而终端节点集合为 $Z=(v_1,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{14},v_{15})$. 各非终端节点至下一个非终端节点或终端节点具有不同的行动选择, 而函数 x 安排这些行动的归属. 例如, 行动$\{Y,W,N\}$都是归属于 v_0 节点, $\{o,q\}$归属于 v_2 和 v_3 节点, 而$\{i,j\}$则归属于 v_4, v_5, v_6 以及 v_7 节点. 函数ρ安排每个非终端节点的归属参与者, 即属于参与者 P_1 的非终端节点集合为$\{v_0,v_4,v_5,v_6,v_7\}$, 属于参与者 P_2 的非终端节点集合为$\{v_2,v_3\}$. 函数σ是能够实现图 1 博弈树的关键, 它能建立一个非终端节点与其一个行动达到新节点的继承关系, 如图中 v_0 节点选择了行动 W 后到达了 v_2 节点. 每个终端节点底下都写有参与者达到该终端节点能获取的实际收益, 我们在最前面使用 u_i 来表示参与者 i 能获得的收益. 除此之外, 博弈定义的最后一个属性 I 的含义是参与者所不能控制的信息集, 即图中虚线范围内的多个选择节点同属一个信息集, 每个信息集都已经写上其所属的参与者. 参与者无法分辨一个信息集中的非终端节点. 例如, 在参与者 P_1 做出行动选择后, 参与者 P_2 无法得知此时它处于 v_2 节点还是 v_3 节点, 因为其无法得知参与者 P_1 选择了哪个行动.

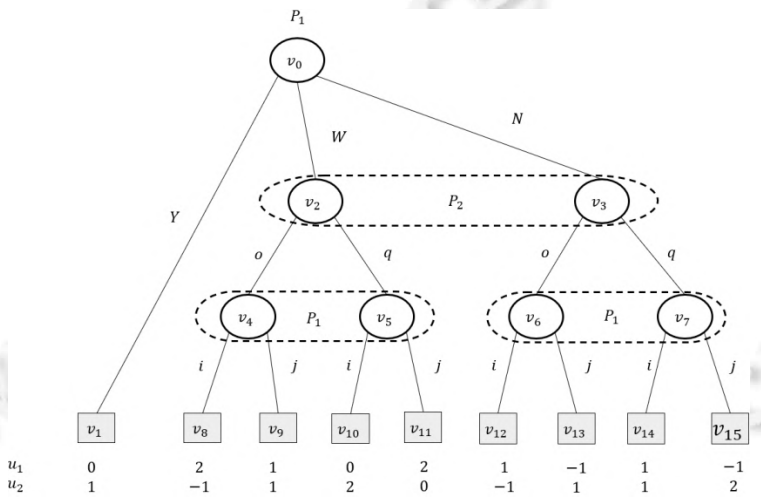

图 1 博弈树的示意图

策略决定了参与者在博弈的任何阶段采取的行动. 根据文献[26], 我们关注的是行动策略. 它比纯策略更具通用性, 具体定义如下.

定义 2. 将 G 作为一个博弈, 参与者 i 的策略 s_i 为一个函数. 它为每一个信息集 $I_{i,j}\in I_i$ 选择节点的行动 $x(I_{i,j})$ 分配概率, 并且每个概率之间都具有相互独立的特性. 策略配置文件是一个由全部参与者策略组成的列表, 记为 $s=(s_i)_{i\in N}$. 没有参与者的策略配置文件被定义为: $s_{-i}=(s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_n)$. 可以直接将策略配置文件记为: $s=(s_i,s_{-i})$.

例如, 图 1 中具有两个策略配置文件(s_1,s_2), 其中, $s_1=\left(\left[\frac{1}{3}(Y),\frac{1}{3}(W),\frac{1}{3}(N)\right],\left[\frac{2}{3}(i),\frac{1}{3}(j)\right],\left[\frac{1}{5}(i),\frac{4}{5}(j)\right]\right)$, $s_2=([0(o),1(q)])$. 对于参与者 P_1 来说, 策略 s_1 说明了它在面临信息集 $I_{1,1}$ 时, 选择其中的任何一个行动的概率都是相等的; 面临信息集 $I_{1,2}$ 时, 有 2/3 的概率会选择行动 i, 1/3 的概率会选择行动 j; 而面临信息集 $I_{1,3}$ 时, 只有 1/5 的概率会选择行动 i, 有 4/5 的概率会选择行动 j. 而对于参与者 P_2 来说, 策略 s_2 表示它面临信息集 $I_{2,1}$ 时, 具有百分之百的概率会选择行动 q, 零概率会选择行动 o.

2.2 智能合约

除了需要分析参与者面临的博弈和策略以外, 方案还需要处理它们选择之后引起的实际利益变化. 一个直接的解决思路是, 引入可信第三方运行奖惩机制以保证博弈游戏的公平执行. 然而, 伴随着第三方的引入, 该方案也会面临着新的安全挑战, 如, 如何确保第三方真的可信, 保证可信第三方在处理争议时, 不会偏袒任

何一方，避免可信第三方发生偷窃参与者资金等恶意行为．为此，本文提出使用智能合约替代可信第三方处理双方利益，并且原先的可信第三方将作为一个威慑实体存在，只在双方出现争议时才会被智能合约唤醒辅助其判断恶意方的工作，以减少方案对可信第三方的依赖．

智能合约的概念由 Nick Szabo 于 1995 年提出[31]，其定义为：“一个智能合约是一套以数字形式定义的承诺，包括合约参与方可以在上面执行这些承诺的协议”．本质上来说，智能合约是一段程序，它以计算机指令集的形式描述人设置的承诺协议．相对于由人来执行的传统合约形式，智能合约交由机器执行，能够在一定程度上规避人恶意执行导致的作弊行为．因为对于机器来说，它是根据指令执行程序，并不会主观主动地进行作恶，只有在被人控制时被迫地强制作弊．然而，机器的运行极其容易被人控制，这导致智能合约缺少能支撑其运行的环境．所以，智能合约在被提出的十几年里都只是一个理论的概念，并没有被实现．直到 2009 年，Satoshi Nakamoto 提出了比特币[32]，引起了区块链的发展热潮．区块链能够促使不可信的用户在一个环境下按照规定诚实地执行同一个任务，这为智能合约提供了契合的运行环境．如今，智能合约已经从理论概念跨越到实际应用，并且其作用受到了广泛的好评和肯定，诸如以太坊[33]、币安智能链[34]等热门区块链都支持智能合约．

智能合约的运行可以分为两个阶段：智能合约的部署和智能合约的调用．本文以以太坊中智能合约的运行作为例子，介绍这两个阶段的大致流程．在以太坊系统中存在着一个以太坊虚拟机 EVM，其作用如沙盒一样将合约执行的环境与区块链隔离开[35]．以太坊上的各个节点将由高级编程语言编写的智能合约转码为 EVM 指令集，并最终在本地部署的 EVM 虚拟机上以堆栈的形式执行每一条指令生成对应的状态变化．以太坊上智能合约的部署流程如图 2 所示．首先，编译器将用户编写的 solidity (以太坊智能合约所支持的图灵完备编程语言)代码编译成为 EVM 字节码；其次，用户调用以太坊的 API，将 EVM 字节码加入到交易数据中，形成以太坊交易，并将该交易公布在以太坊网络中；然后，接收到该交易的其他以太坊节点将验证有效性，通过后将其加入到交易池中，并最终将其加入到新区块中上链；最后，当包含该部署智能合约的以太坊交易被确认后，用户可以通过以太坊 API 查询到该合约的地址，以进一步调用它．智能合约的地址会公布在以太坊上，因此不只是用户，以太坊上的任何一个用户都可以调用该智能合约．调用智能合约的过程可以简单地描述为：一个以太坊用户使用以太坊 API 调用对应地址的智能合约，并输入相关参数以及函数名，然后节点将生成对应的以太坊交易，用户将其公布在以太坊网络中等待矿工执行确认即可．

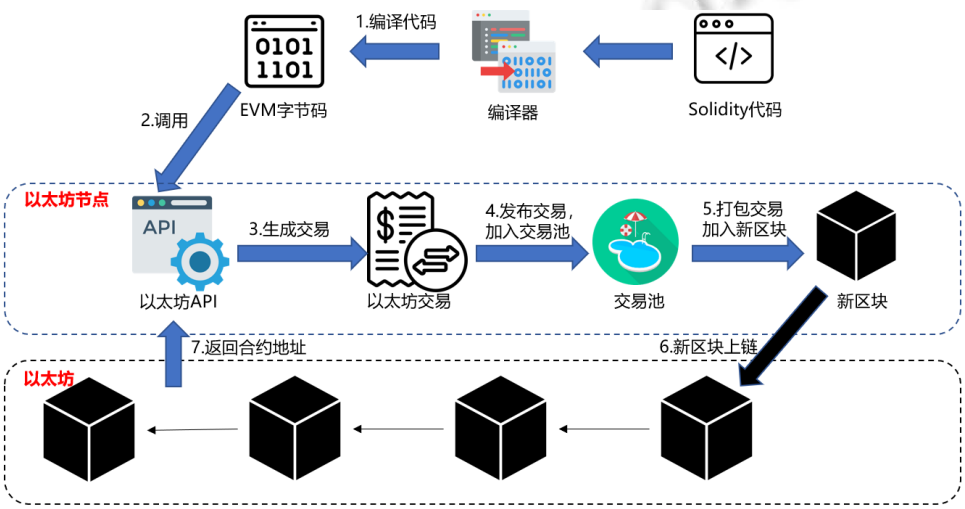

图 2　智能合约部署示意图

3 系统模型和威胁模型

本节首先介绍本文依据的工业互联网外包数据公开审计系统模型, 以及其中存在的威胁. 最后再详细介绍本文方案想要达成的设计目标.

3.1 系统模型

图 3 为本文的系统模型. 该系统中有 5 个实体, 分别为用户、审计者、云存储服务器、智能合约和可信第三方. 其中, 用户作为其他四者的雇佣方, 智能合约作为审计者和云存储服务器的工作判决者, 可信第三方只会在审计者和云存储服务器出现争议时才会被唤醒. 各个实体的具体介绍如下.

- 用户: 用户是工业互联网数据的提供方. 为了节省数据存储的开销, 用户雇佣云存储服务器外包存储工业互联网数据. 同时, 为了确保自己外包的数据不会被不可信的云存储服务器恶意删除, 其雇佣审计者对外包的数据进行定时的审计. 然而, 审计者也是不可信的, 为此, 用户部署智能合约以抵御双方的恶意行为.
- 云存储服务器: 云存储服务器是提供外包存储服务的供应商. 本文假设云存储服务器具有理性且不可信的特征, 即其以自己的利益优先, 并可能为了更高的利益做出恶意行为.
- 审计者: 审计者是出售算力的实体, 其被用户雇佣使用算力为用户做审计服务. 同时, 审计者也具有理性且不可信的特征, 也有可能为了利益做出恶意行为.
- 智能合约: 智能合约为区块链的一个应用, 在本系统中作为下发审计任务、支付审计者和云存储服务器薪资以及处罚双方恶意行为的用户代理者. 由于区块链的性质, 智能合约不会受任何一方控制. 因此, 从双方的角度上, 智能合约就像是一个公正的裁决者.
- 可信第三方: 可信第三方(trusted third party, TTP)协助智能合约检验审计证据. 当审计者和云存储服务器产生争议时, TTP 就会被唤醒并重新执行审计协议, 给出正确的审计证据以协助合约判决.

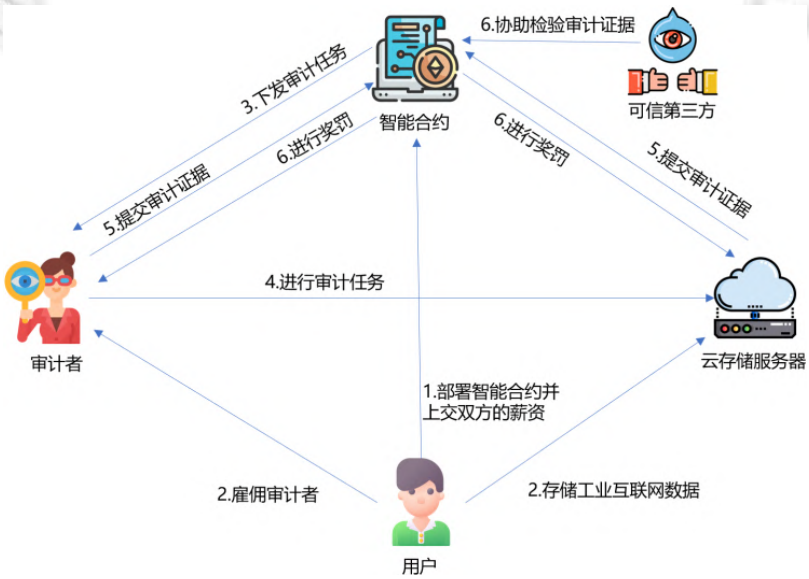

图 3 系统模型图

在清楚上述各实体在系统里的角色后, 根据图 3, 即可得出本系统的大致流程.

- 首先, 用户需要部署智能合约, 在智能合约上设置审计任务和奖罚机制, 并提交用于支付审计者和云存储服务器的薪资.
- 然后, 用户会把需要存储的数据提交给云存储服务器, 通知审计者审计相关的信息, 并要求双方都

在智能合约上注册.

- 智能合约会下派审计任务给审计者, 审计者根据任务审计云存储服务器上的数据.
- 审计者和云存储服务器最后会把审计得到的审计证据提交到智能合约上, 智能合约对审计证据进行检验: 如果正确, 则支付薪资给双方; 反之, 则唤醒 TTP 协助判决, 智能合约根据判决结果对双方进行奖惩.

3.2 威胁模型

威胁模型是指系统模型中实体可能发生的不诚实或侵犯另外实体利益的行为. 此处, 本文只考虑审计者和云存储服务器的威胁模型, 用户、智能合约以及可信第三方并没有作恶的动机. 对审计者和云存储服务器威胁模型的具体分析阐述如下.

- 审计者威胁模型: 审计者的动机是获取更大的利益, 因此, 其可能会不诚实执行审计流程, 返回任意结果作为审计证据, 以节省审计所消耗的算力成本. 或者, 审计者可以发起延迟审计攻击, 以节省一定的审计开销. 除此之外, 任何可以获取更高利益的行为审计者都会接受, 即使侵犯了用户的利益.
- 云存储服务器威胁模型: 云存储服务器的动机也是为了获得更高的利益, 因此, 它也可能会不诚实执行审计流程, 返回任意的结果给用户. 同时, 为了获得更高的收益, 云存储服务器可以删除用户的数据, 将获得的额外空间转租给其他用户. 然而, 为了不让用户发现它的作弊行为, 其必须能够通过审计者的审计. 为此, 云存储服务器可以根据审计者理性的特点发起恶意合谋攻击, 贿赂审计者一起欺骗用户.

值得注意的是, 在审计者威胁模型中, 本文没有考虑审计者发起恶意合谋攻击的情况. 这是因为审计者发起合谋的动机没有云存储发起合谋的动机强烈, 并且云存储服务器接受审计者合谋邀请的概率可忽略不计. 所以, 本文只考虑云存储服务器会发起恶意合谋.

3.3 设计目标

基于系统模型和威胁模型, 我们从效率、安全性和可拓展性这 3 个方面定义了本文的设计目标, 具体内容如下所示.

- 效率: 方案所涉及的算法应具有高效性, 即不应该对运行的环境有过高的要求, 算法运行效率高. 同时, 各方在调用智能合约时, 只需要较低的运行费用.
- 安全性: 本文提出的方案应该满足以下安全要求:
  1) 公平性: 审计者和云存储服务器能够获得的利益不应该被对方影响, 即不存在由于对方的行为而影响到自己利益损失的情况, 双方各自能获得的利益只取决于自己做出的行为.
  2) 抵御威胁: 方案应该能够抵御威胁模型中定义的各种安全威胁.
- 可拓展性: 本文提出的方案应该具备可拓展性, 其体现在方案中的完整性审计部分具有可替换性, 即只要是满足一定条件的公开审计方案, 就都能应用到我们的方案里.

4 具体方案

本节详细介绍本文提出的基于博弈合约的安全高效工业互联网外包数据公开审计方案. 该方案包括两个重要组成部分: 完整性审计和博弈合约. 需要注意的是, 本文方案中的完整性审计部分具有可替换性, 任何一个满足条件(审计者和云存储服务器能生成一个可比较的审计证据)的可公开审计方案都能应用到我们的方案中.

4.1 完整性审计

完整性审计是指审计者、云存储服务器和智能合约三者执行用户指定的工业互联网外包存储数据审计任务. 这里, 我们使用 Ateniese 等人[8]提出的经典 PDP 方案作为本方案的完整性审计部分, 以简单地介绍我们方案对完整性审计部分的要求, 任何满足要求的公开审计方案都可以替换该完整性审计部分. 如图 4 所示, 完整

性审计部分可分为初始化阶段、查询阶段、证明阶段和验证阶段.

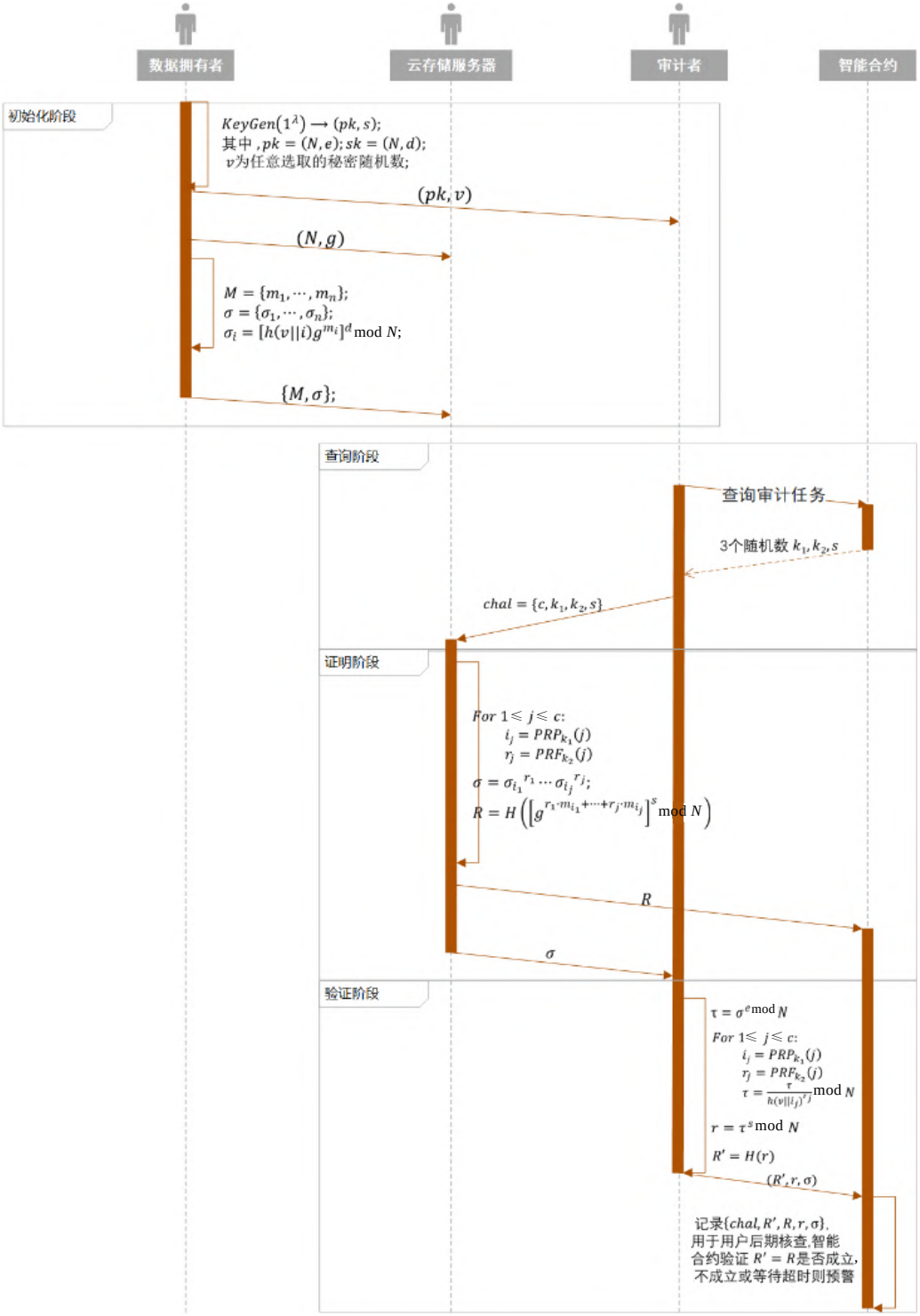

图 4 基于博弈合约的公开审计流程图

各阶段的详细描述如下.

a) 初始化阶段: 用户调用 $KeyGen(1^k)$函数生成公私钥对(pk,sk). 其中, k 为一个安全参数. 具体地, 用户选取两个大素数 p 和 q, 并计算 $N=pq$. 选取一个随机的整数 e, 其中, e 与$(p-1)(q-1)$互质, 并计算 $d=e^{-1} \bmod (p-1)(q-1)$. 则 $pk=(N,e)$, $sk=d$. 然后, 用户将自己需要存储的工业互联网数据进行分块并生成标签. 具体地, 将数据拆分为 n 块, 即$(m_1,m_2,..,m_{n-1},m_n)$; 接下来, 选择一个随机数 v, 对每一块的数据生成标签$\sigma_i(1\leqslant i\leqslant n)$, 即 $\sigma_i=[H(v\|i)g^{m_i}]^d \bmod N$. 其中, i 为数据块的编号, g 为 QR_N 群的一个生成元. 最后, 用户设定审计中每一轮查询的数据块数 $c(c<<n)$, 并在区块链上部署智能合约(囚徒合约), 上交用于支付审计者和云存储服务器的薪资. 用户通知审计者和云存储服务器分别提交押金 d_a 和 d_s 以及公钥 PK_a 和 PK_s 完成注册(不失一般性, 本文假设审计者和云存储服务器采用与用户相同的密码体系, 这里不再赘述). 在双方注册完后, 用户将(pk,v)作为验证密钥发送给审计者, 将(N,g)和处理后的工业互联网数据发送给云存储服务器.

b) 查询阶段: 审计者根据用户的要求, 在一定时间内调用囚徒合约上的查询审计任务函数, 开始一轮审计任务. 囚徒合约会生成 3 个随机数(k_1,k_2,s)作为该轮审计的参数, 并将它们发送给审计者. 其中, k_1 是伪随机置换函数 $PRP(\cdot)$的密钥, k_2 是伪随机函数 $PRF(\cdot)$的密钥, s 是用于掩盖查询数据明文信息的随机数. 然后, 审计者根据这些随机数生成质询 $chal=(c,k_1,k_2,s)$, 并使用自己的私钥 SK_a 对查询请求进行签名, 即 $\sigma_a=sign_{SK_a}(chal)$, 其中, $sign(\cdot)$为双方约定的签名函数(例如基于 RSA 的签名). 最后, 审计者把$(chal,\sigma_a)$发送给云存储服务器.

c) 证明阶段: 云存储服务器验证审计者发送的签名无误后, 云存储服务器根据查询生成被查询数据的聚合签名和审计证据. 具体地, 首先, 云存储服务器根据 k_1 使用 $PRP(\cdot)$生成被查询的 c 个数据的索引, 并根据 k_2 使用 $PRF(\cdot)$生成 c 个随机数, 即 $i_j=PRP_{k_1}(j)$, $r_j=PRF_{k_2}(j)\ (1\leqslant j\leqslant c)$. 然后, 云存储服务器计算被查询数据的聚合签名$\sigma$和审计证据 R, 其中, $\sigma=\sigma_{i_1}^{r_1}...\sigma_{i_j}^{r_j}$, $R=H([g^{r_1\cdot m_{i_1}+...+r_j\cdot m_{i_j}}]^s \bmod N)$. 最后, 云存储服务器把审计证据 R 发送给囚徒合约, 使用私钥 SK_s 对聚合签名σ进行签名, 即$\sigma_s=sign_{SK_s}(\sigma)$, 将$(\sigma,\sigma_s)$发送给审计者.

d) 验证阶段: 审计者先验证云存储服务器的签名σ_s, 确认无误后, 使用验证密钥对聚合密钥进行验算, 得到$\tau'=\sigma^e \bmod N$, 再根据 k_1 和 k_2 与 $PRP(\cdot)$和 $PRF(\cdot)$得到被质询数据的索引 i_j 和随机数 $r_j\ (1\leqslant j\leqslant c)$. 然后, 审计者进一步运算, 得到 $\tau=\dfrac{\tau'}{H(v\|i_j)^{r_j}} \bmod N$, 并计算审计证据 $R'=H(r)$, 其中, $r=\tau^s \bmod N$, 再把(R',r,σ)发送给囚徒合约. 最后, 囚徒合约对比 R 和 R'的值: 如果一致, 则相信双方都诚实地执行审计流程; 反之, 则唤醒可信第三方进行验证, 根据验证结果奖励惩罚作弊的一方. 除此之外, 囚徒合约把(r,σ)记录在区块链上, 便于用户后期审核.

值得注意的是, 在完整性审计流程中所提到的智能合约特指囚徒合约. 该合约基于囚徒困境博弈问题设计, 即签订的双方能够通过合作来获得最大的利益, 但是双方无法达成这种合作, 最后, 各方都只能做出诚实的行为以获取较为高的利益. 然而, 在智能合约的场景下, 囚徒困境中所提到的双方合作能够被达成. 仅需有一方构建一个促成双方合作的合谋合约, 该合谋合约需要满足的条件是双方都必须提交押金, 并且该押金的金额要大于另一方在囚徒困境中因为背叛而损失的金额. 为了实现本文方案的安全性, 我们详细分析了双方在囚徒合约以及合谋合约中的利益关系, 并从中发现破绽, 设计了背叛合约打破双方因为合谋合约所建立的信任关系, 从而迫使双方诚实审计. 具体分析如第 4.2 节所示.

4.2 博弈合约

在本节中, 我们分析上述完整性审计流程中审计者和云存储服务器的利益关系, 并得出双方的博弈树, 进而设计博弈合约. 博弈合约主要由囚徒合约、合谋合约以及背叛合约组成. 其中, 囚徒合约为审计者和云存储服务器主要执行的合约; 而合谋合约和背叛合约并不会被理性的双方执行, 它们是为了确保双方会诚实地

执行囚徒合约而设计. 接下来, 我们会详细介绍每个合约对应的博弈树, 以凸现它们的作用. 博弈树会牵涉到审计者和云存储服务器提交的押金以及用户支付的薪酬等金额, 为方便理解, 我们先进行统一定义, 如下所示.

- b: 合谋时, 云存储服务器给审计者的贿赂金额.
- c: 审计者审计需要付出的成本.
- f: 争议发生时, 唤醒 TTP 需要支付的金额.
- d_s: 在囚徒合约中, 云存储服务器需要提交的押金.
- D_a: 在囚徒合约中, 审计者需要提交的押金.
- D_c: 在合谋合约中, 云存储服务器和审计者需要提交的押金.
- w: 用户给审计者的薪资.
- h: 用户给数据存储服务器的薪资.
- j: 用户对审计者举报行为的奖励, 等于 d_s+d_a+w-f.
- g: 云存储服务器发起合谋能够得到的额外金额.
- z_a: 一个中间变量, 等于 $w-c+d_s-f$.
- z_s: 一个中间变量, 等于 $h+d_a-f$.
- d_c: 在合谋者合约中, 审计者和云存储服务器需要提交的押金.
- T_1: 用户要求审计者必须在该时间之前调用查询审计任务函数, 对应完整性审计流程中查询阶段的开始时间.
- T_2: 云存储服务器需要回应的最晚时间, 对应完整性审计流程中证明阶段的结束时间.
- T_3: 审计者提交审计证据的最后时间, 对应完整性审计流程中验证阶段的结束时间.

同时, 上述变量之间还存在以下关系.

- $w>c$, 用户支付的薪资必须大于审计者审计的成本, 否则审计者不会进行审计.
- $d_s>c+f$, 云存储服务器提交的押金必须要大于审计的成本与唤醒 TTP 所需要的花费之和, 否则, 即使云存储服务器违背了囚徒合约, 审计者也不能获得更高的利益.
- $d_a>f$, 审计者提交的押金要大于唤醒 TTP 所需要的花费, 否则, 即使审计者违背囚徒合约, 云存储服务器也不能获得更高的利益.
- $d_s>g>b$, 云存储服务器获得的额外收益要大于其给出的贿赂金额, 否则没有合谋动机. 同时, 云存储服务器在囚徒合约上提交的押金要大于额外收益, 否则, 其可以选择恶意作弊获取额外收益.
- $f>w$, 唤醒 TTP 需要的费用要高于雇佣审计者的费用, 否则, 用户可以直接要求 TTP 进行完整性审计.

4.2.1 囚徒合约

囚徒合约由用户部署, 审计者和云存储服务器双方签署进行调用. 审计者和云存储服务器之间的关系在囚徒合约中形成了囚徒困境. 双方都被要求在合约上提交押金, 当任何一方出现恶意行为时, 其押金会被支付给诚实的另一方. 通过这样的奖惩机制, 可以迫使理性的参与者诚实地执行审计. 囚徒合约的描述如下.

- 囚徒合约由用户部署, 用户需要设置每次审计的数据块数 c 以及提交审计者和云存储服务器的薪资 w 和 h. 值得注意的是, 用户可以一次提交多份薪资, 以支撑一段时间内审计工作的花费.
- 囚徒合约由审计者和云存储服务器签署, 双方都需要分别提交押金 d_a 和 d_s 以及公钥 PK_a 和 PK_s.
- 审计者需要在 T_1 时间内调用囚徒合约的查询审计任务函数, 否则将被智能合约进行惩罚.
- 云存储服务器需要在 T_2 时间内回应审计者和在囚徒合约上提交审计证据, 否则其押金将被扣除.
- 如果审计者在 T_3 时间内提交最终的审计证据:
  - 如果双方提交的审计证据一致, 囚徒合约将分别支付双方的薪资 w 和 h.
  - 如果双方提交的审计证据不一致, 囚徒合约将唤醒 TTP, 并根据 TTP 验证的结果扣除作弊一方的押金, 该押金支付完 TTP 的唤醒费用后, 剩下的部分发送给诚实的另一方. 如果双方提交的

审计证据都错误, 则它们的押金被全部扣除.

- 如果审计者没有在 T_3 的时间内提交最终的审计证据, 因徒合约会扣除它的押金.

囚徒合约要求双方提交公钥, 是为了避免任何一方发起的恶意干预审计流程行为. 当审计者故意发送错误的查询或云存储服务器故意发送错误的查询数据聚合标签后, 诚实的一方可以使用对方发送的签名作为证据上诉囚徒合约, 以惩戒对方. 囚徒合约最终的判断审计结果方法简单地设置为对比双方的审计证据, 这是由于此时双方处于囚徒困境里面, 它们做出的最好的行动就是诚实地执行审计流程. 对于双方具体的行为和利益分析如图 5 所示, 为博弈 1 的博弈树, 即审计者和云存储服务器在囚徒合约的博弈树.

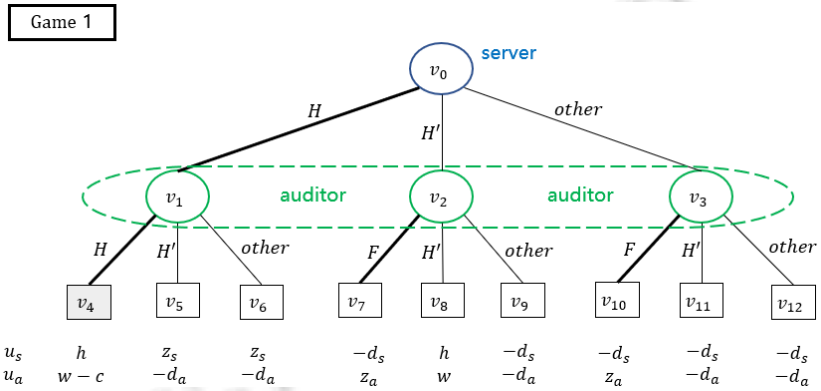

图 5 囚徒合约博弈树示意图

在图 5 所描述的博弈 1 中, 一共有两个参与者, 即 P={*server*,*auditor*}. 用户不考虑作为参与者, 这是由于用户只部署合约以及提交双方的薪资, 并不参与审计流程. 云存储服务器可以采用的行动集合为 A_s={H,H', *other*}, 审计者可以采用的行动集合为 A_a={H,H',F,*other*}, 其中, H 代表参与者提交正确的审计证据, H'代表参与者提交协商得到的虚假审计证据, F 代表参与者提交一个举报的证据, *other* 代表参与者提交其他任意结果的错误审计证据. 两个参与者在该博弈中都只有一个信息集, 即 $I_{s,1}$={v_0}和 $I_{a,1}$={v_1,v_2,v_3}. 它们具体的收益显示于最下面两行变量(u_s,u_a).

接下来, 我们将证明在该博弈中, 双方最终都会诚实地执行完整性审计流程, 并发送正确审计证据 H, 到达双方当前的利益最大点 v_4. 即云存储服务器和审计者此时的策略分别为:

$$s_s^1=([1(H),0(H'),0(other)]) \text{ 和 } s_a^1=([1(H),0(H'),0(F),0(other)]).$$

从审计者即 *auditor* 的角度来看, 当其处于非终端节点 v_1 时, 它面临着 3 个选择的行动 H, H'和 *other*, 而各个行动到达的终端节点 v_4, v_5, v_6 的收益分别是(w–c), ($-d_a$)和($-d_a$), 因此它会选择诚实地执行审计流程, 从而发送审计证据 H 以达到最大收益的终端节点. 当其处于非终端节点 v_2 时, 它面临着 3 个选择的行动 F, H'和 *other*, 而各个行动到达的终端节点 v_7, v_8, v_9 的收益分别是(z_a), (w)和($-d_a$), 因此它会选择诚实地执行审计流程, 从而发送举报证据 F 以达到最大收益的终端节点. 值得注意的是, 当云存储服务器与审计者协商发送一个虚假审计证据 H'时, 审计者可以背叛云存储服务器而上传举报证据 F, 从而获得更高的收益 z_a. 当其处于非终端节点 v_3 时, 它面临着 3 个选择的行动 F, H'和 *other*, 而各个行动到达的终端节点 v_{10}, v_{11}, v_{12} 的收益分别是(z_a), ($-d_a$)和($-d_a$), 因此它会选择诚实执行审计流程, 从而发送举报证据 F 以达到最大收益的终端节点. 即审计者无论在哪个非终端节点, 它的最优行动都是诚实地执行审计流程. 在云存储服务器的视角, 它采用行动能到达的终端节点为 v_4, v_7 和 v_{10}, 而其中收益最高的节点为 v_4, 所以云存储服务器也会诚实地执行审计流程, 以发送正确的审计证据 H.

4.2.2 合谋合约

上述分析了审计者和云存储服务器在囚徒合约上最大收益的行动. 然而, 我们仔细观察图 5 发现, 双方最大收益的终端节点并不是 v_4 而是 v_8, 但双方在囚徒合约上无法达成, 即双方无法信任对方达成共识. 这也是

囚徒困境中的一个典型特征, 囚徒们可以做出合谋的恶意行为, 以获得与诚实执行所得收益相比更高的收益. 然而, 在传统的囚徒困境中, 理性的两个囚徒无法在困境中达成合谋的收益点. 智能合约的存在打破了这个困境, 双方的信任前提可以使用智能合约来达成, 从而达到困境中的最大收益点. 这就是合谋合约的作用,审计者和云存储服务器可以通过签署合谋合约建立信任, 从而双方都返回一个相同的虚假审计证据 H'给囚徒合约, 欺骗用户获得更大的利益. 本文从合谋发起者云存储服务器的角度设计了相关的合谋合约, 并分析双方合谋的利益情况, 为后续打破合谋做准备. 合谋合约的描述如下.

- 合谋合约由云存储服务器部署, 其需要在合谋合约上提交贿赂金 b, 以达到增大审计者合谋动机的目的.
- 合谋合约由云存储服务器和审计者共同签署, 双方都需提交相同金额的押金 d_c. 该押金的金额大于任意一方在囚徒合约中损失的最高值($d_c>d_a+d_s+w-c+h$), 为双方的合谋提供保障, 实现双方的信任.
- 审计者和云存储服务器应该在 T_2 之前完成合谋合约的签署, 超过时间后合约签署失败, 已经提交的押金会被合约退回.
- 合谋合约在 T_3 对比双方提交的审计证据.
  - 如果双方都返回虚假审计证据 H', 合谋合约将 d_c 退回给云存储服务器, d_c+b 退回给审计者.
  - 如果任何一方诚实地返回了审计证据 H', 而另一方违背合谋合约, 那么诚实方会收到 $2d_c+b$, 而作弊的另一方失去全部金额.
- 若双方都返回了其他结果的审计证据, 那么合谋合约将 d_c+b 退回给云存储服务器, d_c 退回给审计者.

合谋合约的签署发生在云存储服务器回应审计者和囚徒合约之前, 这是因为合谋达成后, 云存储服务器可以删除用户的数据, 将得到的存储空间转租给其他用户, 以获得额外的收获 g, 而此时, 云存储服务器已经无法运行完整性审计的证明阶段. 合谋合约将打破囚徒合约所营造的囚徒困境, 双方在合谋合约的作用下, 可以合作欺骗用户达到囚徒困境中利益最大点. 为了避免这种恶意的行为, 我们对双方在合谋合约中的博弈进行分析, 并尝试从中找出它们的薄弱点, 以击破它们的合谋. 如图 6 所示, 为博弈 2 的博弈树, 即审计者和云存储服务器在合谋合约中的博弈树.

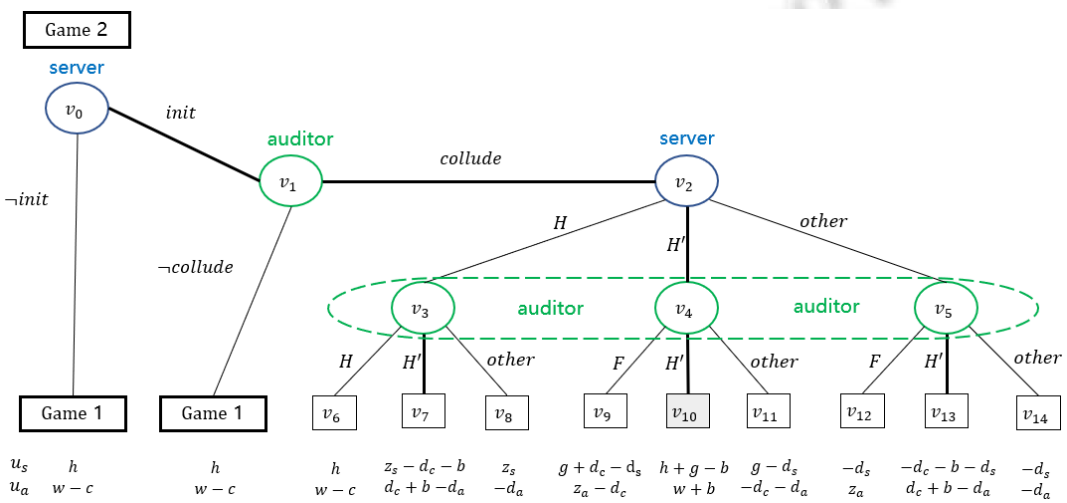

图 6 合谋合约博弈树示意图

在博弈 2 中, 仍然存在两个参与者 P={*server*,*auditor*}, 但是它们可做出的行动集合与博弈 1 的并不相同, 分别为 A_s={*init*,¬*init*,*H*,*H*′,*other*}和 A_a={*collude*,¬*collude*,*H*,*H*′,*F*,*other*}. 云存储服务器可以选择初始化部署合谋合约或者不初始化, 而审计者也可以选择参与合谋或者不参与合谋. 审计者(*audit*)具有两个信息集, 分别为 $I_{a,1}=\{v_1\}$, $I_{a,2}=\{v_3,v_4,v_5\}$; 云存储服务器(*server*)也具有两个信息集, 分别为 $I_{w,1}=\{v_0\}$, $I_{w,2}=\{v_2\}$. 双方的收益情况

在图 6 底部, 值得注意的是, 博弈 2 在博弈 1 的基础上分析, 即博弈 2 中双方的收益情况也是在博弈 1 的基础上进行修改.

接下来, 我们将证明在该博弈中, 双方最终能到达利益最大的终端节点为 v_{10}. 双方策略为:

s_a^2 ={[1(*collude*),0(¬*collude*)],[0(*H*),1(*H′*),0(*F*),0(*other*)]}和 s_s^2 ={[1(*init*),0(¬*init*)],[0(*H*),1(*H′*),0(*other*)]}.

首先, 从审计者的角度来思考, 其会做出的最佳行动. 由于合谋能达到囚徒困境中最大的利益点, 所以审计者会选择参与合谋, 即在非终端节点 v_1 选择行动 *collude*. 然后, 我们从其他非终端节点继续分析审计者的行动, 当审计者位于非终端节点 v_3 时, 它能到达终端节点 v_6, v_7 和 v_8, 而其对应的收益大小情况为 $v_7(d_c+b-d_a)>v_6(w-c)>v_8(-d_a)$, 所以审计者会选择行动 H', 即上交 H'. 再从非终点节点 v_4 和 v_5 上思考, 审计者最高利益的行动还是上交 H', 因此审计者在博弈 2 中的最佳行动为(*collude*,*H′*). 最后, 从云存储服务器的视角出发, 发起合谋会使它获得更高的利益, 所以它在非终端节点 v_0 处会选择行动 *init*. 由于审计者做出的行动一定是 H', 云存储服务器能达到的最终端点只能为 v_7, v_{10} 和 v_{13}, 而它们所对应云存储服务器收益最高的点为 $v_{10}(h+g-b>z_s-d_c-b)>-d_c-b-d_s)$, 因此, 云存储服务器会采取的最大收益行动为(*init*,*H′*). 综上所述, 理性的双方最终会达到最大收益的终端节点为 v_{10}, 也即双方都会返回一个虚假相等的审计结果 H'给囚徒合约.

4.2.3 背叛合约

博弈 2 的出现打破了博弈 1 原本要实现的囚徒困境, 双方由于能获取更高的利益而发起恶意合谋行为. 因此, 我们可以从双方的最终能获取的利益出发, 设计方法打破它们之间的合谋. 具体地, 给审计者提供一种方式, 使其背叛云存储服务器可以获得更高的利益, 即背叛合约. 背叛合约允许审计者汇报云存储服务器发起的合谋, 然后根据云存储服务器提交的审计证据来判断合谋是否存在, 存在则用户会支付审计者额外奖励. 值得注意的是: 背叛合约是审计者和用户私下签署的, 因此云存储服务器无法得知审计者是否做出了背叛的行为, 审计者可以继续提交虚假的审计证据 H', 不仅可以获取贿赂金 b, 还不会被用户扣除囚徒合约的押金并获得云存储服务器在囚徒合约上的押金. 背叛为审计者提供了更高的利益, 而对于云存储服务器, 它害怕审计者背叛自己, 导致其造成大损失, 所以云存储服务器并不会发起合谋, 双方合谋的信任被打破. 背叛合约的描述如下所示.

- 背叛合约由审计者部署, 审计者和用户共同签署.
- 审计者需要在背叛合约上提交押金 f, 如果审计者恶意部署背叛合约, 该押金将会被支付给 TTP, 作为唤醒它的费用. 同时, 审计者还会在背叛合约上提交云存储服务器选取的 H'.
- 用户提交用于支付给审计者的奖励 j.
- 背叛合约需要审计者和用户在 T_3 之前签署, 超过时间后合约签署失败, 提交的押金都会被退回.
- 背叛合约在 T_3 会唤醒 TTP, TTP 将检验审计者和云存储服务器提交的审计证据.
  - 如果云存储服务器提交的审计证据是 H', 背叛合约将 $f+j$ 转给审计者, TTP 唤醒的费用从云存储服务器提交的囚徒合约押金中扣除.
  - 如果云存储服务器提交的审计证据是其他值, 说明是审计者进行的恶意举报, 背叛合约将 f 转给 TTP, j 转给用户, TTP 唤醒的费用由审计者提交的背叛合约押金支付.

背叛合约充分利用审计者是理性的性质, 对云存储服务器产生威慑, 打破了审计者和云存储服务器由于合谋合约建立起来的信任. 云存储服务器害怕损失囚徒合约的押金, 而不敢部署合谋合约, 没有合谋合约的存在, 背叛合约也不会被审计者部署, 所以只会剩下囚徒合约被部署和签署. 至此, 在背叛合约的作用下, 审计者和云存储服务器无法打破囚徒困境, 最终双方只能回到博弈 1 的关系, 诚实地执行完整性审计, 以谋取最大利益. 图 7 为背叛合约所对应的博弈 3 的博弈树.

在博弈 3 中, 拥有两个参与者, 即审计者和云存储服务, 参与者集合为 P=(*auditor*,*server*). 审计者的行动集合 A_a={*Report*,¬*Report*,*H*,*H′*,*F*,*other*}, 云存储服务器的行动集合 A_s={*H*,*H′*,*other*}. 其中, 审计者的 *Report* 行动指其是否上报云存储服务器的合谋行为, 由于博弈 3 在博弈 2 的基础上开展, 所以如果审计选择不上报合谋行为, 双方将会回到博弈 2 中. 审计者的信息集有两个, 分别为 $I_{a,1}$={v_0}和 $I_{a,2}$={v_2,v_3,v_4}, 云存储服务器

只有一个信息集，为 $I_{s,1}$={v_1}. 各个终端节点下面的两行数据为双方对应的实际收益.

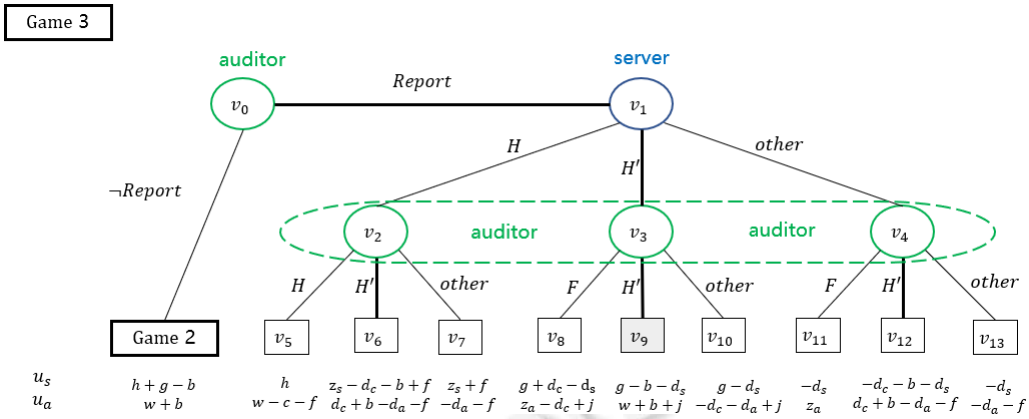

图 7 背叛合约博弈树示意图

接下来，我们将证明在博弈 3 中，双方最终会到达终端节点 v_9. 即:

- 审计者对应采取的策略为 s_a^3 {[1(*Report*),0(¬*Report*)],[0(*H*),1(*H′*),0(*F*),0(*other*)]};
- 云存储服务器对应采取的策略为 s_s^3 ={[0(*H*),1(*H′*),0(*other*)]}.

首先，从审计者的视角来考虑其在博弈 3 中的行动. 如图 7 所示，审计者一开始会位于非终端节点 v_0 做出选择，由于作为背叛者能获得更高的利益，所以它会选择活动 *Report* 上报对手合谋行为. 其次，从云存储的视角，它并不知道审计者会做出背叛行为. 根据博弈 2 的分析可得出，云存储服务器在合谋合约的约束下肯定会选择提交虚假审计证据 *H′*. 最后，审计者达到非终端节点 v_3，而其面临的 3 个终端节点(v_8,v_9,v_{10})中，v_9 节点对应的收益最大，所以它会选择行动 *H′*，即提交虚假审计证据 *H′*. 综上所述，在博弈 3 中，双方最终都会到达终端节点 v_9，审计者会采取的行动为(*Report*,*H′*)，云存储服务器会采取的行动是(*H′*).

从总体上来看，在审计者和云存储服务器都是理性的情况下，3 个合约的相互制衡关系使得最终只有囚徒合约会被部署和签署. 因此，博弈合约的设计保证了囚徒合约运行的安全性，防止了审计者和云存储服务器的合谋攻击，使双方一直处于囚徒困境下，而囚徒合约本身就能抵御审计者的延迟审计攻击和恶意不审计行为，即博弈合约确保了本文方案的安全性.

除此之外，本文提出的方案对于参与审计流程的三方来说都具备接受和使用的动机.

- 首先，对于用户来说，其目的是保护存储在云存储服务器上的数据，所以其通过雇佣审计者进行完整性审计以保护数据的安全. 然而，审计者与云存储服务器会产生恶意的行为侵犯其利益，因此，用户更愿意选择具有更高计算开销但更为安全的审计方案. 而本文方案既能为原有高效审计方案提供额外的安全性能，又能为用户节省大量的运行开销，所以用户具有较强的动机选择使用本文方案.
- 其次，对于审计者来说，本文方案并没有为其增加额外的计算开销，并且审计者还有可能会获得额外的经济收益. 具体来说，审计者可以通过举报恶意的云存储服务器从而获得额外的收益. 因此，理性的审计者也会愿意使用本文方案.
- 最后，对于云存储服务器来说，本文方案没有引入较为复杂的密码学工具，因此其不需要在本地提供或部署特殊的环境；同时，本文方案的执行效率高，能够进一步节省云存储服务器协助审计流程所需的花费. 因此，理性的云存储服务器也会选择使用本文的方案.

5 安全性分析

在本节中，我们将介绍本文提出的方案如何实现设计目标中定义的安全性. 分别从公平性与抵御威胁两个方面分析.

5.1 公平性

在方案的完整性审计部分, 审计者和云存储服务器的公平性威胁是它们可能收到对方故意设置的错误信息, 而导致它们无法正常地完成审计流程被扣除押金. 为此, 囚徒合约上专门设置了上诉函数解决这个威胁, 当任何一方因为对方发送了错误的信息而导致审计失败时, 其可以调用囚徒合约的上诉函数, 提交对方发送的信息以及签名. 囚徒合约会审核上诉方提交的签名, 确认是对方生成的后, 雇佣 TTP 按照上诉信息重新执行一遍完整性审计流程, 根据执行结果进行惩罚. 上诉函数的设置, 使得理性的审计者和云存储服务器不会恶意地干预对方执行审计流程. 至此, 方案完整性审计部分的公平性已经被实现.

在方案的博弈合约部分, 双方的公平威胁来源于提交审计证据时可能会被对方发现, 从而对方根据博弈分析寻找自己收益最大的行动. 然而, 本方案中双方提交的信息先经过加密, 然后再经由囚徒合约解密. 因此, 任何一方都无法获取对方发出的审计证据. 即使自己的选择被对方知道, 对方也没有能力发起恶意攻击侵犯自己的利益. 由每一个智能合约的博弈分析可得: 对于理性的审计者和云存储服务器, 它们只能做出固定的行动以获取最大利益. 综上所示, 此方案博弈合约部分的公平性也被实现.

综上, 本文提出的基于博弈合约的安全高效工业互联网外包数据公开审计方案的公平性可以得到保障.

5.2 抵御威胁

审计者威胁模型主要是指审计者可能会发起偷懒执行审计攻击、延迟审计攻击以及合谋攻击. 本文的方案可以抵御以上各种威胁, 分析如下.

- 对于审计者发起的偷懒执行审计攻击, 囚徒合约可以避免. 在囚徒合约的作用下, 审计者和云存储服务器的关系变为竞争者, 因此, 审计者如果偷懒不执行审计, 而云存储服务器执行了审计, 那么审计者就会被惩罚失去押金. 由此可见, 理性的审计者为了自己的利益, 不会偷懒而是诚实地执行审计.
- 对于审计者发起的延迟审计攻击, 囚徒合约能够避免. 囚徒合约要求审计者在一定的时间内调用查询审计任务函数, 即审计者必须在一定的时间内进行审计, 不然, 审计者的押金也会被扣除. 因此, 理性的审计者不能发起延迟审计攻击.
- 对于审计者受到贿赂而参与的合谋攻击, 方案提出的 3 个博弈合约能够避免. 根据博弈合约的设计, 审计者和云存储服务器除了诚实地执行囚徒合约以外, 没有其他获取最大利益方法. 因此, 理性的审计者不会与云存储服务器发起合谋攻击.

云存储服务器威胁模型主要是指云存储服务器可能会发起偷懒执行审计攻击和恶意合谋攻击. 本文的方案可以抵御云存储服务器威胁, 分析如下.

- 对于云存储服务器发起的偷懒执行审计攻击, 本方案的抵御方法和抵御审计者的偷懒执行审计攻击一样. 由于云存储服务器和审计者在囚徒合约里是竞争者关系, 理性的云存储服务器为了自己的利益, 只能放弃偷懒诚实地执行审计.
- 对于云存储服务器发起的恶意合谋攻击, 本方案博弈合约部分可以抵御. 云存储服务器可以通过部署合谋合约来达成它和审计者合作的信任, 然而审计者可以在接收合谋后部署背叛合约获得更高的利益. 理性的云存储服务器知道背叛合约的存在后, 为了防止自己损失囚徒合约的押金, 它并不会发起合谋. 双方没有合谋合约建立的信任, 无法进行合谋. 最后, 双方能够诚实地执行囚徒合约以谋求最大的利益.

综上所述, 本文提出的基于博弈合约的安全高效工业互联网外包数据公开审计方案能够抵御上述定义的恶意威胁.

6 性能分析

本节进行方案的性能测试与分析: 首先, 我们对方案博弈合约部分的合约运行效率进行测试, 以便体现我们方案的适应性; 接下来, 我们会将方案与其他抗参与者恶意行为的方案进行对比, 以证明本文方案在对

原有审计方案运行效率产生可忽略不计影响的前提下, 为其提供额外的安全性.

6.1 合约Gas开销

Gas 是智能合约运行所消耗的燃料, 因此, 其花费也作为智能合约运行效率的评判值. 我们首先使用 Truffle Suite 旗下的 Ganache 环境在本地搭建私有区块链[36], 以测试我们的 3 个智能合约的功能. 我们本地测试主机的 CPU 为 Intel(R) Core(TM) i7-10875H CPU@2.30 GHz, 内存为 32 GB, 操作系统为 Windows 11. 其次, 在确认其功能无误后, 我们部署 3 个智能合约在以太坊的公有测试链 Ropsten 上, 以测试其运行的 Gas 花费. 最后, 我们再根据统计以太坊 Gas 价格的网站 ETH Gas Station 显示的实时平均 Gas 价格, 计算合约每个函数对应的运行 Gas 花费. 我们测试的日期是 2021 年 11 月 11 日, 当日平均的 Gas 价格为 112 Gwei, Gas 费的计算公式为 $Gas_{fee}=Gas_{price}\cdot Gas_{cost}$. 具体的数据见表 1.

表 1 方案智能合约运行开销表格

合谋名称	函数名称	Gas_{cost}	Gas 花费(ether)
囚徒合约	部署合约	2 410 598	0.270 0
	Init	70 044	0.007 8
	Register	52 116	0.005 8
	Task	91 169	0.010 2
	Deliver	62 686	0.007 0
	Pay	38 203	0.004 3
	Check	39 385	0.004 4
	Response	58 632	0.006 6
合谋合约	部署合约	1 330 689	0.149 0
	Init	51 649	0.005 8
	Join	55 496	0.006 2
	Compare	83 791	0.009 4
背叛合约	部署合约	804 788	0.090 1
	Init	50 909	0.005 7
	Arouse	55 518	0.006 2
	Conduct	45 866	0.005 1

其中, 需要首先说明的是智能合约中每个函数的功能. 在囚徒合约中, *Init* 是用户调用提交审计者和云存储服务器的薪资并设置相关参数的函数, Register 是审计者和云存储服务器提交押金的注册函数, Task 是审计者调用的查询审计任务函数, Deliver 是审计者和云存储服务器提交审计证据的函数, Pay 是根据双方审计证据进行奖罚的函数, Check 是用户产生怀疑唤醒 TTP 审查双方提交审计者的函数, Response 是 TTP 返回结果的函数. 在合谋合约中, Init 是云存储服务器提交押金并设置相关参数的函数, Join 是审计者提交押金参与合谋的函数, Compare 是合谋合约对比双方最终提交审计证据并奖罚的函数. 在背叛合约中, Init 是审计者提交押金进行设置的函数, Arouse 函数是用户调用提交押金的函数, Conduct 是判断云存储服务器是否发起合谋的函数.

综上所示, 审计者和存储服务器每次审计调用的函数应该是囚徒服务器中的 Register, Task, Deliver 和 Pay 函数, 而 Init 函数用户一般只调用一次存储足够的资金, 其他函数在审计者和存储服务器是理性的情况下将不会被调用. 因此, 我们的方案一轮审计下来的消耗总 Gas 为 244 174, 总 Gas 花费为 0.027 3 ether. 本方案的智能合约 Gas 花费较少, 具有较高的实用性. 然而, 由于以太币的货币属性, 其价格极其不稳定. 例如 2021 年 11 月以来, 其单价已经突破 2 万人民币. 因此, 部署与执行合约的经济开销(Gas 花费·以太币价格)也会随之增加. 为了解决这个问题, 可以将本方案在联盟链或私有链上实施.

6.2 合约时间开销

智能合约代码应该具备高适应性, 即对矿工运行配置的要求低. 接下来, 我们对方案中智能合约函数的运行时间进行测试, 以进一步凸显本方案智能合约的高适应性.

图 8 为本方案在本地测试主机搭建的 1.8 版本 JDK 环境下的测试结果. 我们主要测试审计者和云存储服务器审计次数与合约执行时间的关系. 具体来说, 审计者审计流程需要执行的囚徒合约函数 Register、Task 和 Deliver 执行 10 次只需 26.5 μs, 而云存储服务器审计流程执行的囚徒合约函数 Register、Deliver 和 Pay 执行

10 次也只需 23.7 μs. 甚至执行多次审计后, 审计者平均每次审计只需 3.65 μs, 即图 8 中较高的虚线; 云存储服务器平均每次审计只需 3.5 μs, 如图 8 较低的虚线所示. 综上所述, 本文方案智能合约代码也具有高效性, 且对运行环境的要求低, 适应性强.

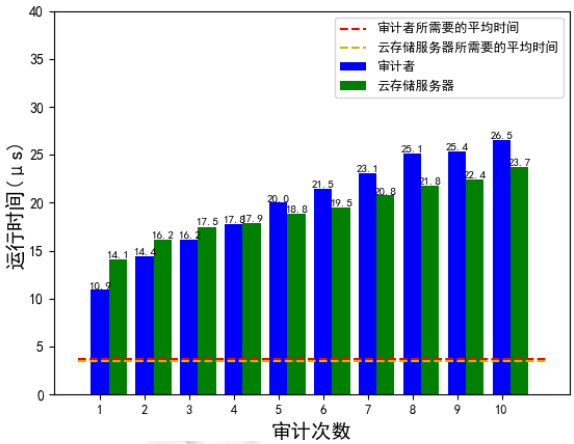

图 8 智能合约时间开销示意图

6.3 方案对比

图 9 所示为我们在本地测试主机上, 利用 1.8 版本 JDK 环境测试的方案效率对比结果. 其中, 本文方案实现了第 4 节所描述的具体方案例子. 除此之外, 本文还实现了 Yang 等人[9]提出的 DPOS 方案、Zhang 等人[30]提出的 CPVPA 方案以及 Ateniese 等人[8]提出的原始 PDP 方案.

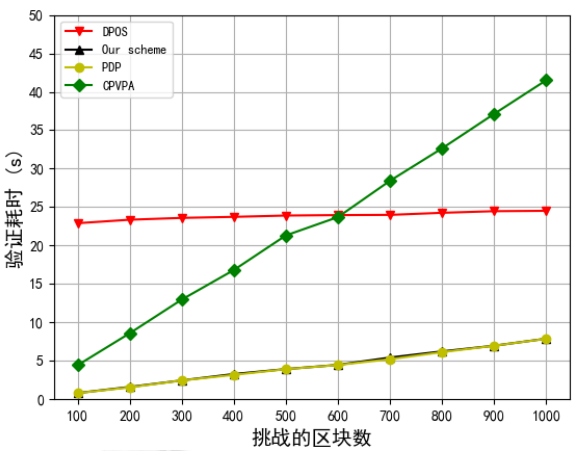

图 9 方案对比示意图

需要注意的是, 本方案所使用的公开完整性审计方案可以被其他公开审计方案可替换. 方案的博弈合约部分通过将原有完整性审计方案的验证过程部署在博弈合约上, 以为其提供抵御参与者恶意行为的安全性. 因此, 在此处, 我们主要对比了各方案的审计效率. 而其余阶段, 本文未对其进行额外的设计, 所以在此不作对比测试. 在图 8 中, 我们方案的审计效率曲线与 Ateniese 等人提出的 PDP 方案审计效率曲线重合, 说明了我们方案与原有 PDP 方案的审计效率基本一致. 这是因为我们所设计的智能合约运行效率高效, 如第 6.2 节所示, 审计者平均审计一次只需 3.65 μs, 云存储服务器审计一次只需 3.5 μs, 智能合约执行只需要微秒级别的时间, 相比原有 PDP 方案执行耗时的秒级别可忽略不计. 而 Zhang 等人提出的 CPVPA 方案审计效率曲线处于我们方案曲线上方, 这是由于其引入了无证书签名作为防范审计者恶意行为的工具, 所以其审计开销会比

PDP 的更大. Yang 等人提出的 DPOS 方案审计效率曲线在挑战区块数较少时比 CPVPA 方案的效率曲线高,这是由于其包含了双线性映射以及多项式承诺等密码学原语, 以至于它们审计的初始开销会比 CPVPA 的更高. 然而, 其引入了批量验证的方法, 能够使得方案审计任意数量的质询数据块都保持同样的开销, 导致了其审计效率曲线不会随着挑战区块数的增加而发生大量的增长. 甚至在挑战区块数到达一定程度后, 其审计效率会比 CPVPA 方案的验证效率更低. 这种方法在一定程度上能够有效地降低审计花费.

总体来说, DPOS 方案以及 CPVPA 方案放弃了一定的效率性以提供了更高的安全性, 它们的效率都比 PDP 要低. 现有的依赖复杂密码工具实现抵御参与者恶意行为的完整性审计方案都有这个特性. 然而, 本文的方案使用了另一种解决思路, 利用基于博弈论的智能合约迫使参与者诚实地执行审计任务. 并且, 实验数据表明, 利用本文方案的完整性审计方案能够在保持原有效率的情况下增加额外的安全性.

7 总 结

在本文中, 我们提出了一个基于博弈合约的安全高效工业互联网外包数据公开审计方案. 该方案为满足一定条件的公开审计方案增加抵御延迟审计攻击和合谋攻击的能力. 方案利用博弈论分析在公开审计过程中审计者和云存储服务器的利益关系, 从而进一步设计能够使它们诚实地执行审计的3个智能合约. 特别地, 方案中使用的公开审计方案可以被其他公开审计方案替换, 能够抵御攻击的原理是精心设计的博弈合约, 其使审计者和云存储服务器位于囚徒困境中, 双方只能通过诚实执行审计才可以获得最大利益. 同时, 本文还通过一系列的测试表明, 所提出的方案具有高效性和可拓展性, 能够高效、安全地实现工业互联网数据公开审计.

References:

[1] Yu XH, Liu M, Jiang XH, Yin YP, Yang X, Liu DF, Zhang HS, Liu XM, Chi C. Industrial Internet architecture 2.0. Computer Integrated Manufacturing System, 2019, 25(12): 2983−2996 (in Chinese with English abstract). http://www.cims-journal.cn/CN/10.13196/j.cims.2019.12.001 [doi: 10.13196/j.cims.2019.12.001]

[2] Cai CW, Qi YD. The empowerment path of Industry Internet to China's manufacturing industry. Contemporary Economy Management, 2021, 43(12): 40−48 (in Chinese with English abstract). [doi: 10.13253/j.cnki.ddjjgl.2021.12.006]

[3] Khan WZ, Rehman MH, Zangoti HM, Afzal MK, Armi N, Salah K. Industrial Internet of things: Recent advances, enabling technologies and open challenges. Computers & Electrical Engineering, 2020, 81: 1−13.

[4] Khan A, Din S, Jeon G, Piccialli F. Lucy with agents in the sky: Trustworthiness of cloud storage for Industrial Internet of things. IEEE Trans. on Industrial Informatics, 2021, 17(2): 953−960.

[5] Xian HQ, Liu HY, Zhang SG, Hou RT. Verifiable secure data deduplication method in cloud storage. Ruan Jian Xue Bao/Journal of Software, 2020, 31(2): 455−470 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5628.htm [doi: 10.13328/j.cnki.jos.005628]

[6] Qiu T, Chi JC, Zhou XB, Ning ZL, Atiquzzaman M, Wu DO. Edge computing in Industrial Internet of things: Architecture, advances and challenges. IEEE Communications Surveys & Tutorials, 2020, 22(4): 2462−2488.

[7] Juels A, Kaliski Jr BS. PORs: Proofs of retrievability for large files. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. New York: Association for Computing Machinery, 2007. 584−597.

[8] Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. New York: Association for Computing Machinery, 2007. 598−609.

[9] Yang AJ, Xu J, Weng J, Zhou H, Wong DS. Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage. IEEE Trans. on Cloud Computing, 2021, 9(1): 212−225.

[10] Xu J, Yang AJ, Zhou JY, Wong DS. Lightweight delegatable proofs of storage. In: Proc. of the Computer Security (ESORICS 2016). Cham: Springer, 2016. 324−343.

[11] Han S, Liu SL, Chen KF, Gu DW. Proofs of retrievability based on MRD codes. In: Huang X, ed. Proc. of the Information Security Practice and Experience. Cham: Springer, 2014. 340−345.

[12] Ren ZW, Wang LN, Wang Q, Xu MD. Dynamic proofs of retrievability for coded cloud storage systems. IEEE Trans. on Services Computing, 2018, 11(4): 685−698.

[13] Zhou L, Fu AM, Mu Y, Wang HQ, Yu S, Sun YX. Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system. Information Sciences, 2021, 545: 254−276.

[14] Fu AM, Li YH, Yu S, Yu Y, Zhang GX. DIPOR: An IDA-based dynamic proof of retrievability scheme for cloud storage systems. Journal of Network and Computer Applications, 2018, 104: 97−106.

[15] Wang Q, Wang C, Li J, Ren K, Lou WJ. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes M, ed. Proc. of the Computer Security (ESORICS 2009). Berlin: Springer Int'l Publishing, 2009. 355−370.

[16] Wu T, Yang GM, Mu Y, Guo FC, Deng RH. Privacy-preserving proof of storage for the pay-as-you-go business model. IEEE Trans. on Dependable and Secure Computing, 2021, 18(2): 563−575.

[17] He DB, Kumar N, Wang HQ, Wang LN, Coo KKR. Privacy-preserving certificateless provable data possession scheme for big data storage on cloud. Applied Mathematics and Computation, 2017, 314: 31−43.

[18] Chen D, Yuan HB, Hu SS, Wang Q, Wang C. BOSSA: A decentralized system for proofs of data retrievability and replication. IEEE Trans. on Parallel and Distributed Systems, 2020, 32(4): 786−798.

[19] Jiang T, Meng WJ, Yuan Xu, Wang LM, Ge JH, Ma JF. ReliableBox: Secure and verifiable cloud storage with location-aware backup. IEEE Trans. on Parallel and Distributed Systems, 2021, 32(12): 2996−3010.

[20] Shen WT, Yu J, Yang GY, Chen XG, Hao R. Cloud storage integrity checking scheme with private key recovery capability. Ruan Jian Xue Bao/Journal of Software, 2016, 27(6): 1451−1462 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4999.htm [doi: 10.13328/j.cnki.jos.004999]

[21] He K, Chen J, Yuan Q, Ji SL, He DB, Du RY. Dynamic group-oriented provable data possession in the cloud. IEEE Trans. on Dependable and Secure Computing, 2019, 18(3): 1394−1408.

[22] Xu J. Auditing the Auditor: Secure Delegation of Auditing Operation over Cloud Storage. IACR Cryptology ePrint Archive, 2011:304, 2011.

[23] Liu XF, Sun WH, Lou WJ, Pei QQ, Zhang YQ. One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage. In: Proc. of the IEEE 2017 Conf. on Computer Communications. 2017. 1−9.

[24] Fan YK, Lin XD, Tan G, Zhang YQ, Dong W, Lei J. One secure data integrity verification scheme for cloud storage. Future Generation Computer Systems, 2019, 96: 376−385.

[25] Zhang Y, Xu CX, Yu S, Li HW, Zhang XJ. SCLPV: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. IEEE Trans. on Computational Social Systems, 2015, 2(4): 159−170.

[26] Dong CY, Wang YL, Aldweesh A, McCorry P, Moorsel AV. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In: Proc. of the ACM SIGSAC Conf. on Computer and Communications Security, Vol.17. New York: Association for Computing Machinery, 2017. 211−227.

[27] Wang C, Chow SSM, Wang Q, Ren K, Lou WJ. Privacy-preserving public auditing for secure cloud storage. IEEE Trans. on Computers, 2011, 62(2): 362−375.

[28] Armknecht F, Bohli JM, Karame GO, Liu ZR, Reuter CA. Outsourced proofs of retrievability. In: Proc. of the ACM SIGSAC Conf. on Computer and Communications Security. New York: Association for Computing Machinery, 2014. 831−843.

[29] Armknecht F, Bohli JM, Karame GO, Li WT. Outsourcing proofs of retrievability. IEEE Trans. on Cloud Computing, 2021, 9(1): 876−888.

[30] Zhang Y, Xu CX, Lin XD, Shen XM. Blockchain-based public integrity verification for cloud storage against procrastinating auditors. IEEE Trans. on Cloud Computing, 2021, 9(3): 923−937.

[31] Szabo N. Formalizing and securing relationships on public networks. First Monday, 1997, 2(9). [doi: 10.5210/fm.v2i9.548]

[32] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2018. https://bitcoin.org/bitcoin.pdf

[33] Vitalik B. Ethereum White Paper. 2013. https://github.com/ethereum/wiki/wiki/White-Paper

[34] Introduction to Binance Chain. https://www.binance.org/

[35] Wohrer M, Zdun U. Smart contracts: Security patterns in the ethereum ecosystem and solidity. In: Proc. of the Int'l Workshop on Blockchain Oriented Software Engineering (IWBOSE). 2018. 2−8.

[36] Anilkumar V, Joji JA, Afzal A, Sheik RS. Blockchain simulation and development platforms: Survey, issues and challenges. In: Proc. of the Int'l Conf. on Intelligent Computing and Control Systems. 2019. 935−939.

附中文参考文献:

[1] 余晓晖, 刘默, 蒋昕昊, 尹杨鹏, 杨希, 刘棣斐, 张恒升, 刘晓曼, 池程. 工业互联网体系架构 2.0. 计算机集成制造系统, 2019, 25(12): 2983−2996. http://www.cims-journal.cn/CN/10.13196/j.cims.2019.12.001 [doi: 10.13196/j.cims.2019.12.001]

[2] 蔡呈伟, 戚聿东. 工业互联网对中国制造业的赋能路径研究. 当代经济管理, 2021, 43(12): 40−48. [doi: 10.13253/j.cnki.ddjjgl.2021.12.006]

[5] 咸鹤群, 刘红燕, 张曙光, 侯瑞涛. 可验证的云存储安全数据删重方法. 软件学报, 2020, 31(2): 455−470. http://www.jos.org.cn/1000-9825/5628.htm [doi: 10.13328/j.cnki.jos.005628]

[20] 沈文婷, 于佳, 杨光洋, 程相国, 郝蓉. 具有私钥可恢复能力的云存储完整性检测方案. 软件学报, 2016, 27(6): 1451−1462. http://www.jos.org.cn/1000-9825/4999.htm [doi: 10.13328/j.cnki.jos.004999]

李涛(1997－), 男, 硕士生, 主要研究领域为区块链应用安全.

翁健(1976－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为公钥密码学, 云安全, 区块链安全.

杨安家(1989－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为物联网安全, 区块链安全, 应用密码学, 数据审计.

郭梓繁(1997－), 男, 硕士生, 主要研究领域为数据审计, 区块链安全.