计算机科学
COMPUTER SCIENCE

CASESC:基于以太坊智能合约的云审计方案

郭彩彩，金瑜

引用本文

郭彩彩，金瑜. CASESC:基于以太坊智能合约的云审计方案[J]. 计算机科学，2023，50(12)：368-376.

GUO Caicai, JIN Yu. CASESC:A Cloud Auditing Scheme Based on Ethereum Smart Contracts[J]. Computer Science, 2023, 50(12): 368-376.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221000185

CASESC:基于以太坊智能合约的云审计方案

郭彩彩　金　瑜
武汉科技大学计算机科学与技术学院　武汉 430065
湖北省智能信息处理与实时工业系统重点实验室　武汉 430065
(guocaicai@ontoweb.wust.edu.cn)

摘　要　云存储凭借其高扩展性、低成本等优点受到广泛关注,但确保云数据的完整性成为了目前亟待解决的问题。由于区块链具有去中心化、不可篡改等特点,可以很好地解决基于第三方审计者的云审计方案中存在的单点失效和安全威胁等问题,因此有学者提出了基于区块链的云审计方案,但这类方案的审计证明均由数据拥有者(DO)或委托其他 DO 进行验证,需要 DO 保持在线状态,加重了审计负担,且绝大部分方案并未在真正的区块链环境中实现。基于此,提出了一种基于以太坊智能合约的云审计方案——CASESC,使用 solidity 语言编写可实现向云服务提供商发起审计请求和验证其返回的审计证明等功能的以太坊智能合约代码,并将审计结果和相关信息记录在以太坊中供 DO 随时查询,使得 CASESC 能代替 DO 完成审计工作,无需 DO 委托验证与实时在线,降低了审计开销。此外,CASESC 分别在以太坊 Goerli 公有链测试网络和 Ganache 搭建的私有链中运行,验证了其可用性。理论分析和实验结果表明,CASESC 可在不增加整体审计开销的情况下大幅降低 DO 的审计开销。

关键词:云审计;区块链;以太坊;智能合约
中图法分类号　TP309.2

CASESC:A Cloud Auditing Scheme Based on Ethereum Smart Contracts

GUO Caicai and JIN Yu
College of Computer Science and Technology,Wuhan University of Science and Technology,Wuhan 430065,China
Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System,Wuhan 430065,China

Abstract　People prefer to use cloud storage due to its advantages of high scalability and low cost,but ensuring the integrity of cloud data has become a security challenge that needs to be solved immediately. While blockchain's characteristics of de-centralization and tamper resistance can greatly solve the problems such as single-point failures and security threats existing in cloud auditing schemes based on third party auditor(TPA),some scholars propose blockchain-based cloud auditing schemes. But these schemes need data owner(DO) or a delegated DO to validate the auditing proof,which not only requires DO to keep online,but increases its auditing burden. Moreover,most of them are only implemented in a simulated blockchain environment. Therefore,this paper proposes a cloud auditing scheme with Ethereum smart contracts—CASESC. CASESC uses solidity language to write Ethereum smart contract code which can send auditing requests and validate the auditing proof returned from cloud server provider(CSP) and stores auditing results and related information in the Ethereum that can be referred to by DO. Without delegating others or keeping online status,CASESC can replace DO to work and reduces its auditing overhead. Besides,CASESC conducts experiments in Ethereum public blockchain called Goerli and private blockchain constructed by Ganache in order to prove its availability. Theoretical analysis and experimental evaluation show that CASESC can significantly reduce the auditing overhead of DO without increasing overall auditing overhead.

Keywords　Cloud auditing,Blockchain,Ethereum,Smart contract

1　引言

云存储是一个以存储和管理数据为核心的云计算系统,其以高效的性能和管理优势[1-2]受到了广泛关注。然而,数据拥有者(Data Owner,DO)对云数据不具备绝对支配权,无法确定云数据的完整性,从而产生了安全和信任问题[3]。例如,云服务提供商(Cloud Server Provider,CSP)可能为节省存储空间而删除不经常访问的数据[4],或因遭受攻击致使内部硬件或软件故障,丢失 DO 所存数据[5],甚至存在窜改数据等恶意行为。

为解决上述问题,有研究者提出了远程数据审计技术,由于当前方案主要依赖于第三方审计者(Third Party Auditor,TPA),这类方案普遍存在中心化、单点失效等缺点[6]。而区块链的去中心化、不可篡改和可追溯等特性可以弥补上述

到稿日期:2022-10-23　返修日期:2023-03-19
通信作者:金瑜(jinyu@wust.edu.cn)

缺点,因此许多基于区块链的云审计方案被提出。但这类方案在移除 TPA 后,其审计验证工作主要由 DO 或委托其他 DO 完成,仍存在以下缺点:1)安全威胁,即被委托的 DO 可能串通 CSP 更改审计证明,欺骗原 DO;2)在线审计,即在审计阶段,DO 需保持在线状态与 CSP 进行交互;3)资源限制,即对于计算资源和通信资源有限的 DO,计算审计挑战和验证审计证明将加重其审计负担。

针对上述问题,本文提出了一种基于以太坊智能合约的云审计方案——CASESC。利用同态验证标签设计审计协议,用 solidity 语言编写智能合约代码,实现发起审计请求和验证 CSP 返回的审计证明等工作,部署好智能合约后生成合约账户供 DO 和 CSP 查询和调用。在此过程中相关的审计信息以交易、日志和事件的形式永久记录在以太坊中以实现溯源,DO 可随时查看智能合约触发的事件查询审计结果,无需 DO 实时在线或委托其他 DO 代为审计,且审计验证工作由智能合约代码自动执行,降低了 DO 的审计开销。CASESC 解决了基于区块链云审计方案中存在的问题。此外,本文分别在以太坊 Goerli 公有链测试网络和 Ganache 搭建的私有链环境中实现了 CASESC,证明了 CASESC 具备一定的有效性和可用性。

2 相关工作

区块链是一种在不依赖任何第三方中心化机构的情况下实现去中心化的点对点交易、协调及协作的技术[7],具有去中心化、不可篡改、可追溯等优势,该技术的兴起推动了远程数据审计技术[8]的发展。通过将挑战和证明等审计交互信息记录在区块链中,来实现公共审计、数据追溯和审计行为追责,解决了引入半可信 TPA 审计方案中存在的各种问题。在结合区块链技术取代半可信 TPA 的方案中,2020 年,Xu 等[9]提出了一种支持仲裁的云审计方案,在 DO 审计失败后利用智能合约和交换哈希技术仲裁数据纠纷。2021 年,Zhang 等[10]结合收敛加密技术和分层角色哈希树提出了支持重复数据删除和数据授权的云审计方案,但该方案在远程审计时需上传 DO 私钥,本地审计需 DO 下载原始数据。同年,Sharma 等[11]提出由 DO 计算并上传默克尔哈希树(Merkle Hash Tree,MHT)和相关数据信息到区块链中,审计时由 DO 计算和对比 CSP 返回的 MHT 根证明,但 CSP 可以访问区块链或保存以前的数据信息来伪造审计证明,无法抵御伪造、重放和删除攻击。随后,Zhang 等[12]提出了多云环境下的云审计方案,设计了一个中心化的 CSP 作为代表参与审计,引入智能合约在 DO 审计失败后进行仲裁并由代表定位出错的 CSP,但中心化的 CSP 代表并不完全可信。

以上工作[9-12]中都完全由 DO 验证审计证明,其中文献[11]还需 DO 额外计算 MHT 等相关审计信息。上述方案对通信资源和计算资源有限的 DO 来说,加大了审计负担。此外,文献[9]和文献[12]在 Go 语言实现的以太坊客户端(Go Ethereum,Geth)中验证了其方案,引入的智能合约仅用于记录审计相关信息,不参与审计工作,文献[10]和文献[11]仅在本地模拟的区块链中验证其方案。

为减轻 DO 的审计负担,2017 年,Liu 等[13]令 DO 或被委托的 DO 对比 CSP 返回的哈希值,证明其和区块链中存储的哈希值是否一致,但该方案利用哈希值作为审计证明无法抵抗伪造攻击和删除攻击。2018 年,Yu 等[14]设计审计区块链结构存储 CSP 上传的审计证明,并由 DO 或被委托的 DO 查询区块链来验证该审计证明。2019 年,Wang 等[15]提出由所有 DO 和 CSP 共同构成区块链网络节点,结合共识机制使每个节点均有权参与审计验证工作,这不仅需要 DO 具备强大的算力,且 CSP 在处理与自己相关的审计信息时可能会伪造区块。2020 年,Huang 等[16]设计由 DO 存储和维护的辅助链表协助审计,且从由多个 DO 构成的群组中选举出组管理者来验证审计证明,这需要 DO 具备额外的存储空间和算力来实时维护辅助链表。同年,Li 等[17]提出将数字签名存储于区块链中,帮助 DO 和 CSP 计算 MHT 根验证证明,以减少审计验证的计算和通信开销,但该方案需委托其他 DO 协助审计,且无法抵抗 CSP 的伪造和删除攻击。2021 年,Shu 等[18]引入去中心化自治组织来减轻恶意实体对审计结果的影响,并结合智能合约协助生成审计挑战,但该方案在委托矿工验证审计证明后,需 DO 对该审计结果进行二次验证。

上述工作[13-18]为降低 DO 的审计开销,选择委托其他 DO 等实体验证审计证明,但被委托的实体可能与 CSP 共谋欺骗原 DO,仍然无法摆脱不诚实的实体行为对审计结果的影响。此外,文献[18]在 Ganache 搭建的以太坊私有链中验证其方案,且引入了智能合约用于生成审计挑战,其余方案均在本地模拟的区块链中验证其方案。

为完全消除不诚实的实体行为对审计机制的影响,结合智能合约具备隔离运行和部署后无法更改等特点,有学者提出利用智能合约[19]代替 DO 进行审计验证工作。2018 年,Renner 等[20]提出将审计函数运行在审计框架节点中,并为每个数据生成独立的智能合约以存储散列值等信息,但该方案无法在 CSP 仅存储数据散列值的情况下保证原数据的完整性。同年,Xue 等[21]提出由 DO 存储并计算根据 m 个随机数序列构造的 m 棵 MHT 结构,审计时由 DO 挑选一组随机数序列发送给 CSP 计算 MHT 根,并将其作为审计证明,且由智能合约进行验证,但该方案仅能支持 m 次审计,且 DO 仍需较高的计算量和存储量。2020 年,Peng 等[22]和 Yuan 等[23]先后提出了基于双线性对的云审计方案,CSP 返回证明时调用智能合约内部的审计函数代替 DO 进行审计验证。2021 年,Xie 等[24]提出由 CSP 定期挑选随机数计算审计证明,并将其返回给智能合约进行验证,以实现 DO 和 CSP 的零交互,但该方案中 CSP 在已知随机数和数字签名的情况下可以伪造审计证明,因此并不安全。

上述工作[20-24]利用智能合约代替 DO 进行审计,避免了其他实体代为审计带来的安全隐患,但上述工作仅在模拟的区块链上实现其方案,其运行在本地的智能合约更容易受到攻击,信息更容易泄露,且不满足区块链的开放性和独立性。

3 预备知识和威胁模型

3.1 同态验证标签

同态验证标签是由 Ateniese 等[25]首先提出的具有同态特征且不可伪造的可验证数据标签,由 DO 计算并在审计

过程中帮助验证数据完整性。其特性如下：

1)无块验证：DO 可以在 CSP 不返回原始数据的情况下验证数据完整性。

2)同态性：对于任意的 b_i 和 b_j，可根据其同态验证标签 $Tag(b_i)$ 和 $Tag(b_j)$ 计算出 b_i+b_j 的同态验证标签 $Tag(b_i+b_j)$。

本文利用基于 RSA 的同态哈希函数[26]构建同态验证标签，秘密选取两个互异的大素数 p 和 q 并计算公开的 RSA 模数 $N=p\cdot q$。设 g 是乘法循环群 G 的一个生成元，满足 $gcd(g,N)=1$。对于数据 b_i，其同态验证标签为 $Tag(b_i)=g^{b_i} \bmod N$。根据模运算的性质可知其满足同态性，证明如下：

$$\begin{aligned} Tag(b_i+b_j) &= g^{b_i+b_j} \bmod N \\ &= (g^{b_i}\cdot g^{b_j}) \bmod N \\ &= (g^{b_i} \bmod N \cdot g^{b_j} \bmod N) \bmod N \\ &= (Tag(b_i)\cdot Tag(b_j)) \bmod N \end{aligned} \tag{1}$$

在云审计方案中，常用的 BLS 签名技术过于依赖有限域中的双线性对运算，导致在相同签名条件下，BLS 签名技术所耗费的时间更长[27]。尽管基于椭圆曲线的数字签名技术在相同的安全性条件下具有较短的密钥长度，但其不适用于单次审计多个数据的场景。而利用基于 RSA 的同态哈希函数构建同态标签具备同态特性和不可伪造性，对于任意大小的数据输入，其对应的数据标签长度都不会超过所选 RSA 模数的长度；且根据其同态性可对数据标签进行聚合操作，支持单次审计多个数据。因此该签名协议适用于在分布式数据存储系统中审计单个用户的多个数据。

3.2 模幂运算

模幂运算[28-29]是公钥密码体制中的重要运算，满足模运算性质，对于给定的底数 g、指数 r、模 N，其模幂运算结果为 $g^r \bmod N$，根据模运算基本性质可知：

$$g^{a+b} \bmod N=(g^a \bmod N\cdot g^b \bmod N) \bmod N \tag{2}$$

$$g^{a\cdot b} \bmod N=(g^a \bmod N)^b \bmod N \tag{3}$$

3.3 威胁模型

本文假设区块链网络足够稳定，DO 和 CSP 之间的通信通道足够安全。在此基础上，提出了 3 种威胁类型：

1)CSP 可能会从部署后的智能合约代码中窃取随机数 $\{r_i,\cdots,r_{i+t}\}$ 或从公共参数 $\{chal_i,\cdots,chal_{i+t}\}$ 中恢复 $\{r_i,\cdots,r_{i+t}\}$，以计算能通过验证的审计证明。

2)CSP 可能会因各种原因损坏或删除 DO 数据，为通过审计验证，CSP 可能会伪造数据 b_i^* 或从公开参数中恢复数据 b_i，以计算能通过验证的审计证明。

3)CSP 伪造审计证明，以通过审计验证。

4 CASESC

CASESC 分为初始化阶段和审计阶段，本节先给出系统模型并概述审计流程，然后从这两个阶段详细介绍 CASESC。

4.1 系统模型

CASESC 的系统模型如图 1 所示，主要包含 DO，CSP 和 SC(Smart Contracts，SC)3 类实体，其定义如下。

1)DO：数据拥有者，具有有限的计算、通信和存储资源，具有将个人数据上传到 CSP 上的需求。

2)CSP：云服务提供商，具备大量的存储空间和计算资源，负责存储 DO 的数据、计算审计证明返回给 SC 进行验证。

3)SC：智能合约，隔离运行在以太坊虚拟机(Ethereum Virtual Machine，EVM)中的一段代码，负责响应 DO 的审计请求向 CSP 发起审计挑战、验证 CSP 返回的审计证明和将审计相关信息存储在以太坊中等工作。

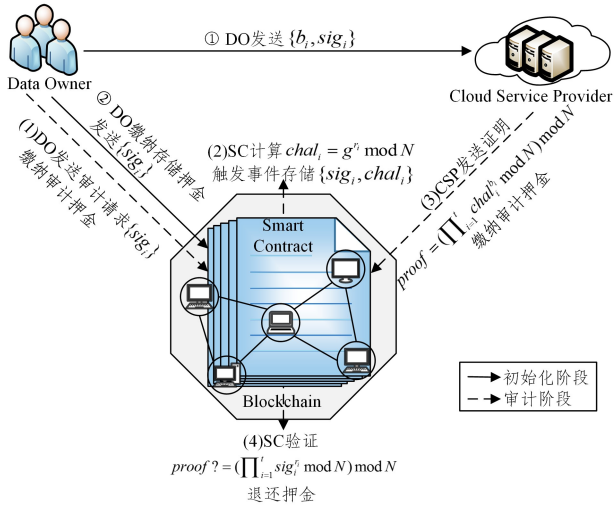

图 1 系统模型

Fig. 1 System model

在 CASESC 中，DO 和 CSP 均有唯一的以太坊外部账号以供发起交易或调用 SC。图 1 给出了 CASESC 的系统流程，在初始化阶段，DO 为每个数据计算数字签名后将数据和签名发送给 CSP，同时 DO 还需发起附带费用的以太坊交易，CSP 根据签名验证数据完整性，成功后存储 DO 数据和签名，并退还 DO 附加费用，否则退还 DO 所有费用。在审计阶段，DO 以交易的形式缴纳审计押金调用智能合约，智能合约暂管押金并向 CSP 发起审计挑战，CSP 以交易的形式附带押金返回审计证明交由智能合约验证。若 CSP 正确存储数据，在扣除 DO 需支付的其他费用后分别返还 DO 和 CSP 的剩余押金；反之，若 CSP 未正确存储数据，将扣除 CSP 的押金以补偿 DO 的损失。整个审计过程都通过交易进行，且调用智能合约时会触发事件以日志的形式存储在以太坊上，可供后续追责。

4.2 初始化阶段

系统初始化时 DO 秘密选取两个互异的大素数 p 和 q(均大于 500 比特)作为保密参数，并计算 RSA 模数 $N=p\cdot q$。g 是乘法循环群 G 的一个生成元，满足 $gcd(g,N)=1$，其中 g 和 N 为系统公共参数。DO 和 CSP 提前商议存储费用、附加费用和押金等规则后加入以太坊，后文将省略费用细节，SC 部署后存储在以太坊中。

DO 分别为 n 个加密后的数据 $\{b_1,\cdots,b_i,\cdots,b_n\}$，$i\in[1,n]$ 计算数字签名 $sig_i=g^{b_i} \bmod N$，并链下传输 $\{b_i,sig_i\}$ 至 CSP。同时，DO 需在链上调用 SC，通过算法 1 发起一笔附带费用的交易，此费用需要比预先商定的存储费用略高，多出来的费用作为附加费用质押在 SC 处，用于帮助 DO 判断 CSP 是否正确接收到 DO 链下发送的数据与签名，最后触发事件 storeSigs 将 $\{sig_i\}$ 存储于以太坊中。伪代码如算法 1 所示。

算法 1 SC 收取费用并触发事件存储 DO 上传的数字签名

```
输入：CSP 地址 CSPAddr，数字签名 sigs
输出：void
1. Begin
2.     require(msg.value ！=0 &&. sigs.length ！=0，“Error!”)；
3.     for i=0 to sigs.length do
4.         emit storeSigs(sigs[i])；
5.         //触发 storeSigs 事件存储签名
6.     end for
7.     CSPAddr.transfer(msg.value-addExpenses)；
8. End
```

CSP 链下收到外包数据 $\{b_i, sig_i\}$ 后监听以太坊事件 storeSigs 获取 DO 上传的 $\{sig_i\}$，比对是否与链下接收的 $\{sig_i\}$ 一致，然后通过公共参数 g 和 N 计算 $sig_i = g^{b_i} \bmod N$ 以判断 DO 是否发送了正确的数据。若上述两次验证均成功，则 CSP 需通过算法 2 发送验证结果 true 并退还 DO 附加费用，随后 DO 可以删除本地数据；若验证失败，CSP 需通过算法 2 发送验证结果 false 并退还 DO 存储费用和附加费用，伪代码如算法 2 所示。

算法 2 SC 退还 DO 费用

```
输入：目标 DO 地址 DOAddr
输出：void
1. Begin
2.     if result==true
3.         DOAddr.transfer(addExpenses)；
4.     else
5.         require(msg.value ！=0，“Error!”)；
6.         DOAddr.transfer(msg.value+addExpenses)；
7.     end if
8. End
```

4.3 审计阶段

DO 共有 n 个数据存储在 CSP 中，审计阶段由 DO 随机选择 t，$t \in [1, n]$ 个数字签名发起审计请求，SC 计算 t 个随机数构造审计挑战，CSP 计算证明后返回给 SC 验证，最后将结果以事件的形式记录在以太坊上，具体步骤如下。

1) $chalGen(sig_i) \rightarrow (chal_i)$

DO 访问事件 storeSigs 挑选 t 个数字签名 $\{sig_i, \cdots, sig_{i+t}\}$，$i \in [1, n]$ 调用 SC 通过算法 3 发起审计请求并提交押金，SC 开启新的审计周期，计算 t 个随机数 $\{r_i, \cdots, r_{i+t}\}$，$r_i \in Z_p$ 和 $chal_i = g^{r_i} \bmod N$，通过触发事件 storeAuditInfos 将 $\{sig_i, chal_i\}$ 记录在以太坊中，其中随机数 r_i 为非公开参数，伪代码如算法 3 所示。

算法 3 SC 向 CSP 发起审计请求

```
输入：目标 CSP 地址 CSPAddr，抽样数字签名 auditSigs
输出：void
1. Begin
2.   require(isAudit==false &&. msg.value ！=0 &&. auditSigs.
length ！=0，“Error!”)
3.   isAudit=true；//开启审计周期
4.   for i=0 to auditSigs.length do
5.     sigiArr[i]=auditSigs[i]；
6.     riArr[i]=getRandNum()；//声明 private
7.     chaliArr[i]=g^riArr[i] mod N；
8.   end for
9.   emit storeAuditInfos(sigiArr，chaliArr)；
10.  depositDO=msg.value；
11.  addrCSP=CSPAddr；
12.  addrDO=msg.sender；
13. End
```

2) $proofGen(sig_i, chal_i) \rightarrow (proof)$

CSP 调用 SC 查询当前审计周期标志，若在审计周期内，则监听以太坊事件 storeAuditInfos 获取 $\{sig_i, chal_i\}$，根据监听获取 sig_i 并查找出对应数据 b_i，根据 $chal_i$ 计算聚合证明为：

$$proof = (\prod_{i=1}^{t} chal_i^{b_i} \bmod N) \bmod N \tag{4}$$

最后调用 SC 返回证明 $proof$ 并提交押金。

3) $verify(proof) \rightarrow (0/1)$

CSP 调用 SC 提交审计证明 $proof$ 和押金后，SC 通过算法 4 进行审计验证，根据内部存储的 r_i 和 DO 发送的 sig_i 计算 $\sigma = (\prod_{i=1}^{t} sig_i^{r_i} \bmod N) \bmod N$，并根据 $proof$ 验证等式 $\sigma ? = proof$，判断 CSP 是否正确存储 DO 数据，数据完整性验证证明如式(5)所示：

$$\begin{aligned} proof &= (\prod_{i=1}^{t} chal_i^{b_i} \bmod N) \bmod N \\ &= (\prod_{i=1}^{t} (g^{r_i} \bmod N)^{b_i} \bmod N) \bmod N \\ &= (\prod_{i=1}^{t} (g_i^{b_i} \bmod N)^{r_i} \bmod N) \bmod N \\ &= (\prod_{i=1}^{t} sig_i^{r_i} \bmod N) \bmod N \\ &= \sigma \end{aligned} \tag{5}$$

若 $\sigma = proof$ 则验证通过，在扣取 DO 需支付的相关存储费用后，SC 需分别退还 DO 和 CSP 的剩余押金。若 $\sigma \neq proof$ 则验证失败，SC 不仅需退还 DO 押金，还需扣取 CSP 押金发送给 DO 作为对 DO 的补偿。最后触发事件 storeProof，storeSigma 和 storeAuditResult 将本次审计结果记录在以太坊中，结束当前审计周期，伪代码如算法 4 所示。

算法 4 SC 验证审计证明

```
输入：审计证明 proof
输出：void
1. Begin
2.     require(msg.value ！=0 &&. isAudit==true，“Error!”)
3.   uint sigma=1；
4.   for i=0 to sigiArr.length do
5.     tempValue=sigiArr[i]^riArr[i] mod N；
6.     sigma*=tempValue mod N；
7.   end for
8.   depositCSP=msg.value；
9.   emit storeProof(proof)；
10.    emit storeSigma(sigma)；
11.    if sigma==proof then
12.      emit storeAuditResult(true)；
13.      addrCSP.transfer(depositCSP+extraExpenses)；
14.    addrDO.transfer(depositDO-extraExpenses)；
15.  else
```

```
16.     emit storeAuditResult(false);
17.     addrDO.transfer(depositDO+depositCSP);
18.   end if
19.   isAudit=false;
20. End
```

5 安全性分析和性能分析

5.1 安全性分析

根据威胁模型可知 CSP 为通过审计验证可能存在不诚实的审计行为，本节针对 3.3 节提出的 3 种威胁类型进行推导和证明，对 CASESC 的安全性进行了理论分析。

定理 1 CSP 无法从 SC 中窃取随机数$\{r_i,\cdots,r_{i+t}\}$或从公开参数$\{chal_i,\cdots,chal_{i+t}\}$中恢复$\{r_i,\cdots,r_{i+t}\}$以计算能通过验证的审计证明。

证明：以太坊智能合约运行在与外界隔离的 EVM 环境中，一旦部署上链则无法更改。本文方案将智能合约代码中随机数$\{r_i,\cdots,r_{i+t}\}$的访问权限设置为 private，只有当前合约内部函数可访问，外部函数和子合约无法访问，且 CSP 无法更改部署后随机数$\{r_i,\cdots,r_{i+t}\}$的访问权限，因此 CSP 无法从部署后的智能合约中获取随机数$\{r_i,\cdots,r_{i+t}\}$。在审计过程中，CSP 已知公开参数有 g，N，$\{sig_i,\cdots,sig_{i+t}\}$和$\{chal_i,\cdots,chal_{i+t}\}$，对于 $chal_i=g^{r_i} \bmod N$，一定存在 $k_i\in Z_p$，使得 $g^{r_i}=k_iN+chal_i$。若 CSP 计算出正确的 k_i，由 DL 假设[30]可知，已知 g 和 g^{r_i} 的值，计算 r_i 是困难的，因此 CSP 无法从$\{chal_i,\cdots,chal_{i+t}\}$中恢复出正确的随机数$\{r_i,\cdots,r_{i+t}\}$。综上所述，CSP 无法窃取或恢复$\{r_i,\cdots,r_{i+t}\}$，来计算能通过验证的审计证明 $proof^*=(\prod_{i=1}^{t} sig_i^{r_i} \bmod N) \bmod N$。

定理 2 CSP 无法伪造数据 $b_j^*\neq b_j$ 或从公开参数中恢复出正确数据 b_j 以计算能通过验证的审计证明。

证明：假设存在数据 $b_j^*\neq b_j$，对于式(6)、式(7)：

$$proof=(\prod_{i=1,i\neq j}^{t} chal_i^{b_i} \bmod N \cdot chal_j^{b_j} \bmod N) \bmod N \tag{6}$$

$$proof^*=(\prod_{i=1,i\neq j}^{t} chal_i^{b_i} \bmod N \cdot chal_j^{b_j^*} \bmod N) \bmod N \tag{7}$$

若要计算出能通过验证的审计证明，则有 $proof=proof^*$，则需满足 $chal_j^{b_j^*}\equiv chal_j^{b_j} (\bmod N)$，根据同余的性质和模幂运算的性质可化简为 $g^{r_j\cdot b_j^*}\equiv g^{r_j\cdot b_j} (\bmod N)$，即 $g^{r_j(b_j^*-b_j)}\equiv 1(\bmod N)$。欧拉函数 $\varphi(N)=(p-1)(q-1)$，且当 $gcd(g,N)=1$ 时，有 $g^{\varphi(N)}\equiv 1(\bmod N)$，根据欧拉定理可知，$r_j(b_j^*-b_j)$是 $\varphi(N)$的倍数。由于保密参数 p 和 q 是大素数，根据大整数分解困难问题可知目前还没有多项式时间算法可以分解大整数 N，且 CSP 无法从 SC 中获取保密参数 r_j，因此 CSP 无法计算出满足条件的 b_j^*。在审计过程中，CSP 已知与数据 b_j 相关的公开参数有 g，N，sig_j，对于 $sig_j=g^{b_j} \bmod N$，一定存在 $k\in Z_p$，使得 $g^{b_j}=kN+sig_j$。若 CSP 计算出正确的 k，由 DL 假设可知，已知 g 和 g^{b_j} 的值，计算 b_j 是困难的，因此 CSP 无法根据公开参数恢复正确数据 b_j。综上所述，CSP 无法伪造数据 $b_j^*\neq b_j$ 或从公开参数中恢复出正确数据 b_j 以计算能通过验证的审计证明。

定理 3 CSP 无法伪造审计证明以通过审计验证。

证明：由定理 1 和定理 2 可知，CSP 无法通过恢复数据 b_i 和 r_i 以计算能通过验证的审计证明。此外，对于公开参数 $sig_i=g^{b_i} \bmod N$ 和 $chal_i=g^{r_i} \bmod N$，一定存在 $k_1,k_2\in Z_p$，使得 $g^{b_i}=k_1N+sig_i$，$g^{r_i}=k_2N+chal_i$。若 CSP 计算出正确的 k_1 和 k_2，由 CDH 假设[31]可知，已知 g^{b_i} 和 g^{r_i} 的值，计算 $g^{r_i\cdot b_i}$ 是困难的，因此 CSP 无法计算出能通过验证的证明 $proof^*=g^{\sum_{i=1}^{t} r_i\cdot b_i} \bmod N$。综上所述，CSP 无法通过伪造证明 $proof^*$ 通过 SC 的验证。

5.2 性能分析

5.2.1 实验配置

本文通过连接 Goerli 测试链和搭建以太坊 Ganache 私有链实现 CASESC。设备参数为 Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz；16.00GB RAM；64 位 Windows10 操作系统。本节用 solidity 语言编写并用 truffle 编译与部署智能合约代码，通过 Infura 创建项目与设置 Goerli 测试链 RPC 接口[1)]连接到 Goerli 测试链搭建公有链环境；通过 Ganache 图形界面软件与设置私有链 RPC 接口[2)]搭建以太坊私有链环境，并用 web3js 库编写 JavaScript 代码实现与智能合约的交互，最后给出源码链接[3)]。以太坊环境配置参数为 nodejs v17.7.2，npm v8.5.2，truffle v5.5.16，ganache v6.12.2，web3 v1.8.0，doten v16.0.3，solc v0.8.17。

5.2.2 实验结果

在基于区块链的云审计方案中，Xu 等的方案[9]（简称 BEAA）和 Zhang 等的方案[12]（简称 BMSA）完全由 DO 完成审计验证工作，并且在 Geth 搭建的以太坊环境中验证了其方案，设计的智能合约仅记录审计相关信息，而不参与审计挑战的生成和验证证明等工作；Shu 等的方案[18]（简称 BDPA）在委托矿工验证审计证明后需由 DO 二次验证该审计结果，减轻了 DO 部分的审计负担，并设计由智能合约生成审计挑战，在以太坊 Ganache 搭建的以太坊私有链中实现其方案。BEAA，BMSA 和 BDPA 均在以太坊环境中实现了其方案，且在不同程度上结合智能合约技术参与审计工作，因此本文选用 BEAA 和 BMSA 在以太坊公有链环境下与 CASESC 作比较，选用 BDPA 在以太坊私有链环境下与 CASESC 作比较，分析在不同区块链环境下，CASESC 与 BEAA，BMSA 和 BDPA 在审计过程中 DO 的时间开销、整体方案的审计总时间开销和 gas 开销。

1)基于以太坊公有链网络

本小节使用 Infura 后端扩展设施连接到以太坊 Goerli

1) https://goerli.infura.io/v3/〈API-KEY〉

2) http://127.0.0.1:7545

3) https://github.com/Belief-ff/Blockchain-BasedCloudAudit.git

测试链网络，而无需运行 Goerli 测试链完整节点。在 Infura 官网注册后创建基于 Web3 网络和 Goerli 测试链的项目后，根据该项目内置的 API-KEY 生成的 RPC 接口[1)]连接到该项目，在用 truffle 编译并部署合约后在控制台用 node 命令运行 web3js 编写的文件实现与智能合约的交互。本节将从审计过程中 DO 的时间开销、审计总时间开销这两个方面分析 BEAA，BMSA 与 CASESC 的性能。

表 1 列出了在以太坊 Goerli 测试链环境下，分别在 100 至 1 000 条数据下进行 10%抽样的审计时间开销的详细结果。由于 Infura 提供了简化以太坊交易管理服务，其通过预先向 Infura 链上存款合约缴纳一些押金并发送中继请求，授权 Infura 代为发送以太坊交易，无需用户持续在线等待交易确认，因此 CASESC 中 DO 的时间开销要远小于以太坊约 15 s 的平均交易时间。

表 1 以太坊公有链各项时间开销

Table 1 Time overhead in Ethereum public blockchain

数据个数	DO 时间开销/ms			总时间开销/ms		
	BEAA	BMSA	CASESC	BEAA	BMSA	CASESC
100	6 999	2 651	911	20 604	22 659	19 568
200	14 146	2 791	917	38 094	38 576	37 171
300	19 949	2 898	924	54 459	52 853	49 141
400	27 920	3 069	932	76 735	79 228	70 100
500	34 268	3 169	940	94 193	105 861	88 171
600	41 071	3 251	946	111 382	131 935	122 962
700	49 124	3 449	954	139 637	154 591	133 588
800	54 389	3 664	960	151 864	164 145	138 189
900	65 990	3 798	969	172 788	183 358	149 486
1 000	70 867	3 956	978	197 393	214 512	172 352

图 2 给出了单个 DO 的时间开销，可以看出在 CASESC 中，DO 的时间开销相比 BEAA 大幅减小，相比 BMSA 减小了一半以上。由于 BEAA 中 DO 一次只能审计一个数据，BMSA 中 DO 需查询 CSP 记录在以太坊智能合约中的聚合证明和聚合签名并验证其正确性，而 CASESC 仅需 DO 在挑战生成阶段挑选并发送 t 个数据签名。以 1 000 条数据进行 10%的抽样审计为例，在本实验中，BEAA 需要 DO 进行 100 次大数模幂运算以分别验证审计证明，BMSA 需要 DO 在访问并获取以太坊审计信息后进行 1 次大数模幂运算和一次哈希运算以验证审计结果，而 CASESC 中 DO 不涉及任何大规模的计算操作。这是在 CASESC 中 DO 时间开销远小于其余两个对比方案的主要原因。

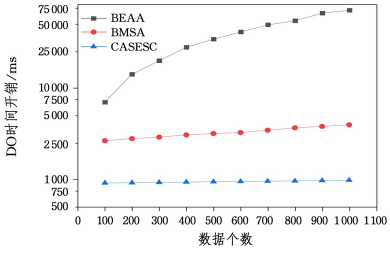

图 2 以太坊公有链中 DO 的时间开销

Fig. 2 Time overhead of DO in Ethereum public blockchain

图 3 给出了在以太坊 Goerli 测试链环境下，单个 DO 分别在 100 至 1 000 条数据下进行 10%的抽样审计时的总时间开销。可以看出 CASESC 的审计总时间开销略小于 BEAA 和 BMSA。由于 BEAA 一次只能审计一个数据，因此累计执行多次审计导致其审计时间开销增大；BMSA 由于引进了中心化 CSP 组织者进行审计任务的分发和审计证明的聚合计算，且聚合前后的审计证明都需要调用智能合约记录在以太坊上，随着审计数据量的增大，其审计时间开销也随之增大。而 CASESC 在 DO 发起审计请求和 CSP 聚合证明后仅各调用一次 SC，造成时间开销增加的主要原因是算法 3 和算法 4 涉及到的 SC 中随机数的计算和证明的验证，同以太坊私有链不同的是，在以太坊公有链中发起交易将存在一定的时延，这也是本文大幅降低了 DO 的审计时间开销，但无法大幅降低总时间开销的主要原因。

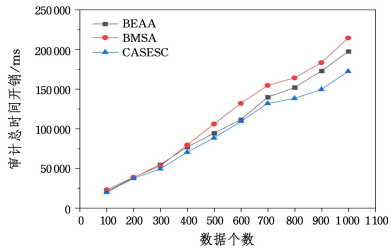

图 3 以太坊公有链中审计总时间开销

Fig. 3 Total time overhead of auditing in Ethereum public blockchain

2）基于以太坊私有链网络

本小节用 Ganache 搭建以太坊私有链结构，并用 Ganache 网络在初始化时分配各有 100 个以太币的 10 个账户作为 DO 和 CSP 的账户参与审计。在控制台用 truffle compile 命令编译智能合约文件后用 truffle migrate 命令将智能合约部署到 Ganache 网络中，truffle 会自动更新智能合约接口，最后在控制台用 node 命令运行 web3js 编写的文件实现与 Ganache 中智能合约的交互。本小节将从审计过程中 DO 的时间开销、审计总时间开销和审计各项操作的 gas 开销这 3 个方面分析 BDPA 和 CASESC 的性能。

表 2 列出了在以太坊 Ganache 搭建的私有链环境下，分别在 100 至 1 000 条数据下进行 10%抽样的审计时间开销的详细结果。与以太坊主网和测试网不同，Ganache 在发起交易时无需同步海量节点，可以快速进行以太坊开发，因此相比公有链测试网络，私有链中 CASESC 的各项时间开销更小。

表 2 以太坊私有链的各项时间开销

Table 2 Time overhead in Ethereum private blockchain

（单位：ms）

数据个数	DO 时间开销		总时间开销	
	BDPA	CASESC	BDPA	CASESC
100	14 539	346	49 075	16 965
200	28 388	352	95 701	34 712
300	43 482	359	144 738	51 309
400	57 951	367	193 000	68 265
500	72 735	373	243 198	83 364
600	86 530	379	294 827	99 156
700	101 826	387	340 469	114 771
800	117 959	391	370 809	131 307
900	131 202	400	398 623	147 379
1 000	144 733	406	445 554	165 344

1) https://goerli.infura.io/v3/〈API-KEY〉

图 4 给出了在以太坊 Ganache 搭建的私有链环境下单个 DO 的时间开销。可以看出 CASESC 极大地降低了 DO 的时间开销，这是因为 BDPA 需要 DO 从抽样数据集合中挑选一个子集进行审计证明的二次验证，DO 需重新计算多次基于乘法群的乘法加法运算和双线性对运算，而 CASESC 仅需 DO 在挑战生成阶段挑选并发送 t 个数据签名，DO 不参与审计证明的验证，且不涉及任何大规模的计算操作。

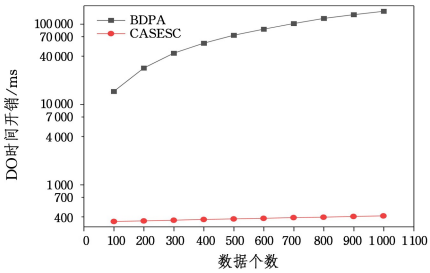

图 4 以太坊私有链中 DO 的时间开销

Fig. 4 Time overhead of DO in Ethereum private blockchain

图 5 给出了在以太坊 Ganache 搭建的私有链环境下，单个 DO 分别在 100 至 1 000 条数据下进行 10%的抽样审计时的总时间开销。可以看出 CASESC 的总时间开销比 BDPA 小，这是因为 BDPA 在审计过程中设计了 campaign 阶段，用于随机挑选一个节点计算挑战随机数，且审计验证阶段在该随机节点对 CSP 返回的证明进行验证后还需 DO 对该审计结果进行二次验证，这些审计流程都加大了 BDPA 的总时间开销。而 CASESC 的审计流程简洁明了，挑战随机数的生成和证明的验证均交由 SC 计算，无需委托，也无需二次验证，因此审计总时间开销更小。

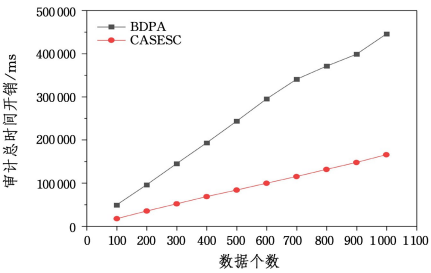

图 5 以太坊私有链中审计总时间开销

Fig. 5 Total time overhead of auditing in Ethereum private blockchain

智能合约运行在以太坊虚拟机 EVM 中，通过 gas 机制限制其执行时间，智能合约操作越复杂，gas 消耗量越大，因此 gas 是衡量以太坊工作量的计量单位。图 6 给出了 CASESC 在 100 条数据下进行 10%的抽样审计过程中各类操作的 gas 消耗情况。一次 a，b，d，f 操作只需 SC 记录一次数据，触发一次事件。由于不涉及大规模的数据运算和处理，因此 a，b，d，f 操作的 gas 开销较小。而 c 和 e 操作需 SC 在 EVM 上进行多次大数模幂运算和大数乘法运算，这导致 c 和 e 操作的 gas 消耗量远大于其他操作。由于 CASESC 利用 SC 进行审计验证工作，其中挑战阶段和验证阶段都有 SC 的参与，因此这两个阶段的 gas 消耗量是最大的。在以太坊系统中 gas 被设计为可以转换为以太币(最大单位为 ether)以支付相关费用，本文撰写期间的 gas 价格为 $1\text{gas}=1.371\,786\,48\times10^{-10}$ ether，由图 6 可知，gas 消耗量最大的两类操作 c 和 e 分别花费 0.000 196 ether 和 0.000 246 ether，这对 DO 和 CSP 来说是可接受的价格。尽管以太币现行价格高昂，但本文为研发用以太坊 DApp 实现云审计机制的机构提供了新思路，因为以太坊 DApp 可以自定义代币价格和数量。

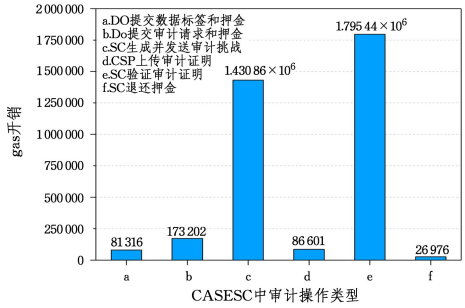

图 6 各项审计操作的 gas 开销

Fig. 6 gas overhead of auditing operations

5.2.3 计算开销与通信开销

由于 BEAA 和 BMSA 均使用同态验证标签技术，因此选用这两个方案同本文方案在计算开销和通信开销这两个方面进行对比分析。为简化表达，使用 C_{me} 表示一次模幂运算的开销，C_{mul} 表示一次乘法运算的开销，C_{mod} 表示一次取模运算的开销，$|b|$ 表示单个数据的大小，$|G|$ 表示群 G 中单个元素的大小，$|p|$ 表示 Z_p 中单个元素的大小。

从表 3 中可以看出，CASESC 中 DO 的计算开销最小，这是因为 CASESC 在初始化阶段为每个数据计算数字签名，仅需计算 n 次模幂运算。而 BEAA 和 BMSA 不仅需要 DO 计算数字签名，在审计阶段还需要 DO 计算审计挑战信息并验证 CSP 返回的审计证明，因此 BEAA 和 BMSA 中 DO 的计算开销更大。

表 3 计算开销对比

Table 3 Computation overhead comparison

方案	DO	SC	CSP
BEAA	$(n+2t)C_{me}$	$2tC_{me}$	$(n+t)C_{me}$
BMSA	$(n+2)C_{me}$	$(t+1)C_{me}+(t-1)C_{mul}+C_{mod}$	$(n+t)C_{me}+(2t-1)C_{mul}+C_{mod}$
CASESC	nC_{me}	$2tC_{me}+(t-1)C_{mul}+C_{mod}$	$(n+t)C_{me}+(t-1)C_{mul}+C_{mod}$

在通信开销方面，因为 SC 将数据存储到以太坊中涉及到共识机制的内容，共识过程中的通信开销难以估算，所以此处仅考虑 DO 和 CSP 的通信开销。在本文方案的实验过程中，$|G|$ 和 $|p|$ 的大小相同均为 258 Bytes，因此从表 4 中可以看出，CASESC 中 DO 的通信开销分别比 BEAA 少 $258(n+t+1)$ Bytes，比 BMSA 少 $258(t+3)$ Bytes，故 CASESC 中 DO 的通信开销最小。

表 4 通信开销对比

Table 4 Communication overhead comparison

方案	DO	CSP
BEAA	$n\|b\|+(n+t)\|G\|+(2n+t+1)\|p\|$	$(n+t)\|G\|$
BMSA	$n\|b\|+(n+3)\|G\|+(n+2t+1)\|p\|$	$(n+5)\|G\|+2t\|p\|$
CASESC	$n\|b\|+(2n+t)\|G\|$	$(n+2t+1)\|G\|$

结束语 本文提出了一种基于以太坊智能合约的云审计方案CASESC,在不增加总审计开销的情况下降低了DO的审计开销,并在以太坊中实现了所提方案。但本文利用的基于RSA的同态验证标签技术中的大数求幂运算十分耗费计算资源,如何设计更高效的求幂算法将是降低本文方案审计开销的关键。此外,本文没有考虑如何制定合理的押金缴纳机制和激励机制来最大限度地调用各实体的审计积极性,也未考虑在审计失败后的错误数据恢复功能,由于DO数据没有备份,因此数据一旦出现问题,对DO而言损失是不可逆的。在后续研究中,将继续对当前工作存在的不足进行改进。

参 考 文 献

[1] ZHANG Y,XU C,LI H,et al. Cryptographic Public Verification of Data Integrity for Cloud Storage Systems [J]. IEEE Cloud Computing,2016,3(5):44-52.

[2] XUE J,XU C,BAI L. DStore:A Distributed System for Outsourced Data Storage and Retrieval [J]. Future Generation Computer Systems,2019,99(1):106-114.

[3] YANG K,JIA X. An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing [J]. IEEE Transactions on Parallel and Distributed Systems,2013,24(9):1717-1726.

[4] ZHANG Y,XU C,LIANG X,et al. Efficient Public Verification of Data Integrity for Cloud Storage Systems from Indistinguishability Obfuscation [J]. IEEE Transactions on Information Forensics and Security,2017,12(3):676-688.

[5] NI J,YU Y,MU Y,et al. On the Security of an Efficient Dynamic Auditing Protocol in Cloud Storage [J]. IEEE Transactions on Parallel and Distributed Systems,2014,25(10):2760-2761.

[6] HAN H,FEI S,YAN Z,et al. A Survey on Blockchain-Based Integrity Auditing for Cloud Data [J]. Digital Communications and Networks,2022,1(1):1-13.

[7] WANG X,ZHA X,NI W,et al. Survey on Blockchain for Internet of Things [J]. Computer Communications,2019,136(1):10-29.

[8] LIU L,XU B. Research on Information Security Technology Based on Blockchain [C]// Proceedings of IEEE 3rd International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). Piscataway:IEEE,2018:380-384.

[9] XUY,REN J,ZHANG Y,et al. Blockchain Empowered Arbitrable Data Auditing Scheme for Network Storage as a Service [J]. IEEE Transactions on Services Computing,2020,13(2):289-300.

[10] ZHANG G,YANG Z,XIE H,et al. A Secure Authorized Deduplication Scheme for Cloud Data Based on Blockchain [J]. Information Processing and Management,2021,58(3):102510.

[11] SHARMA P,JINDAL R,BORAH M D. Blockchain-Based Decentralized Architecture for Cloud Storage System [J]. Journal of Information Security and Applications,2021,62(8):102970.

[12] ZHANG C,XU Y,HU Y,et al. A Blockchain-Based Multi-Cloud Storage Data Auditing Scheme to Locate Faults [J]. IEEE Transactions on Cloud Computing,2021,1(1):1-12.

[13] LIU B,YU X L,CHEN S,et al. Blockchain Based Data Integrity Service Framework for IoT Data [C]// Proceedings of IEEE International Conference on Web Services (ICWS). Piscataway:IEEE,2017:468-475.

[14] YU H,YANG Z,SINNOTT R O. Decentralized Big Data Auditing for Smart City Environments Leveraging Blockchain Technology [J]. IEEE Access,2018,7(1):6288-6296.

[15] WANG H,ZHANG J. Blockchain Based Data Integrity Verification for Large-Scale IoT Data [J]. IEEE Access,2019,7(1):164996-165006.

[16] HUANG P,FAN K,YANG H,et al. A Collaborative Auditing Blockchain for Trustworthy Data Integrity in Cloud Storage System [J]. IEEE Access,2020,8(1):94780-94794.

[17] LI J,WU J,JIANG G,et al. Blockchain-Based Public Auditing for Big Data in Cloud Storage [J]. Information Processing and Management,2020,57(6):102382.

[18] SHU J,ZOU X,JIA X,et al. Blockchain-Based Decentralized Public Auditing for Cloud Storage [J]. IEEE Transactions on Cloud Computing,2021,1(1):1-14.

[19] ZUO W,LO D,KOCHHAR P S,et al. Smart Contract Development:Challenges and Opportunities [J]. IEEE Transactions on Software Engineering,2021,47(10):2084-2106.

[20] RENNER T,MULLER J,KAO O. Endolith:A Blockchain-Based Framework to Enhance Data Retention in Cloud Storages [C]// Proceedings of 26th Euromicro International Conference on Parallel,Distributed and Network-based Processing (PDP). Piscataway:IEEE,2018:627-634.

[21] XUE J,XU C,ZHANG Y,et al. DStore:A Distributed Cloud Storage System Based on Smart Contracts and Blockchain [C]// Proceedings of 18th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP). Cham:Springer,2018:385-401.

[22] PENG F,TIAN H,QUAN H,et al. Data Auditing for the Internet of Things Environments Leveraging Smart Contract [C]// Proceedings of 3rd International Conference on Frontiers in Cyber Security(FCS). Singapore:Springer,2020(1286):133-149.

[23] YUAN H,CHEN X,WANG J,et al. Blockchain-Based Public Auditing and Secure Deduplication with Fair Arbitration [J]. Information Sciences,2020,541(9):409-425.

[24] XIE M,ZHAO Q,HONG H. A Blockchain-Based Proxy Oriented Cloud Storage Public Audit Scheme for Low-Performance Terminal Devices [C]// Proceedings of 21st International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP). Cham:Springer,2021(13155):676-692.

[25] ATENIESE G,BURNS R,CURTMOLA R,et al. Provable Data Possession at Untrusted Stores [C]// Proceedings of 14th ACM Conference on Computer and Communications Security. New York:Association for Computing Machinery,2007:598-609.

[26] GAZZONI F D L,BARRETO P S L M. Demonstrating Data

Possession and Uncheatable Data Transfer [J]. Cryptology ePrint Archive, 2006, 1(1): 150-159.
[27] LIU F, YANG J, LI Z B, et al. A Secure Multi-Party Computation Protocol for Universal Data Privacy Protection Based on Blockchain [J]. Journal of Computer Research and Development, 2021, 58(2): 281-290.
[28] ALIA G, MARTINELLI E. Fast Modular Exponentiation of Large Numbers with Large Exponents [J]. Journal of Systems Architecture, 2002, 47(14/15): 1079-1088.
[29] DESWARTE Y, QUISQUATER J J, SAïDANE A. Remote Integrity Checking [C]// Proceedings of Working Conference on Integrity and Internal Control in Information Systems. Boston: Springer, 2003(140): 1-11.
[30] YAN H, ZHAO F S, SU F G, et al. Quantum Algorithm for Solving Hyperelliptic Curve Discrete Logarithm Problem [J]. Quantum Information Processing, 2020, 19(3): 120-126.
[31] ZHANG F, REIHANEH S N, SUSILO W. An Efficient Signature Scheme from Bilinear Pairings and its Applications [C]// Proceedings of 7th International Workshop on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2004(2947): 277-290.

GUO Caicai, born in 1997, postgraduate, is a member of China Computer Federation. Her main research interests include cloud storage and blockchain.

JIN Yu, born in 1973, Ph. D, associate professor, is a member of China Computer Federation. Her main research interests include cloud computing, peer-to-peer computing and trust model.

（责任编辑：何杨）