XXXX

基于可信执行环境的支持所有权转移的远程数据完整性验证方案

彭溯，冯俐，尹贻然
（海南大学，海南 海口 570228）

摘　要：云存储的普及与数据所有权转移需求的增长，使得支持所有权转移的远程数据完整性验证成为保障云数据安全的关键技术。然而，现有方案普遍存在数据所有者计算负担重、依赖不可信第三方审计者、通信开销大以及缺乏有效的不可否认性机制等问题。为解决上述挑战，提出了一种基于可信执行环境的支持所有权转移的远程数据完整性验证方案。该方案将完整性验证与所有权转移过程中的所有关键计算任务安全地卸载至云服务提供商平台的TEE内执行。基于除TEE的隔离保护外所有实体均不可信的强威胁模型进行了形式化安全分析，并利用Intel SGX开发了原型系统，通过与多种代表性方案进行对比实验以评估其性能。实验结果表明，该方案在标签生成等阶段的计算效率优于对比的方案，计算开销降低幅度在10.1%至86.4%之间，同时实现了云服务提供商与验证者之间的零通信开销。

关键词：远程数据完整性验证；所有权转移；可信执行环境；云存储安全
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.

Remote Data Integrity Checking with Ownership Transfer Based on Trusted Execution Environment

Peng Su, Feng Li, Yin Yiran
Hainan University, Haikou 570228, China

Abstract: The popularization of cloud storage and the growing demand for data ownership transfer have made remote data integrity verification with ownership transfer a key technology for safeguarding cloud data security. However, existing schemes were generally plagued by problems such as heavy computational burden on data owners, reliance on untrusted third-party auditors, high communication overhead, and lack of effective non-repudiation mechanisms. To address these challenges, a remote data integrity verification scheme with ownership transfer based on Trusted Execution Environment (TEE) is proposed in this paper. All critical computational tasks in the integrity verification and

收稿日期：2014−12−01；**修回日期：**2015−01−01
通信作者：彭溯，supeng@hainanu.edu.cn
基金项目：国家自然科学基金资助项目（No. 62362024）；海南大学科研启动基金项目（No. KYQD(ZR)23060）
Foundation Items: The National Natural Science Foundation of China（No. 62362024）；Scientific Research Foundation of Hainan University（No. KYQD(ZR)23060）

ownership transfer are securely offloaded to the TEE of the cloud service provider's platform by our scheme. Formal security analysis is conducted based on a strong threat model where all entities except the isolation protection of TEE are untrusted, and a prototype system is developed using Intel SGX. Comparative experiments with various representative schemes are performed to evaluate its performance. The experimental results demonstrate that this scheme achieves superior computational efficiency in stages such as HVT (Homomorphic Verifiable Tag) generation compared to the benchmark schemes, with computational overhead reductions ranging from 10.1% to 86.4%. Additionally, it achieves zero communication overhead between cloud service providers and verifiers.

Key words: Remote Data Integrity Checking, Ownership Transfer, Trusted Execution Environment (TEE), Cloud Storage Security

0 引言

在信息技术的深度驱动下，数据已成为推动产业变革的核心生产要素。然而，打造具备高安全规格的存储设施，意味着企业需直面高昂的资本开支与复杂的技术挑战。这一现实约束，使得数据托管服务成为众多机构在权衡之后的经济理性选择。尽管云存储简化了数据管理，但也因数据外包引发了多重安全挑战。用户将数据控制权移交云端后，不仅面临传统的数据完整性破坏与意外丢失风险，还需警惕服务商出于商业目的对数据进行的隐性操作，例如根据访问频率自动清理冷数据，或通过减少冗余以降低存储成本。因此，对CSP存储的数据进行完整性验证至关重要。

当前已存在诸多针对云服务器外包数据的有效完整性验证方法[1-8]，无需下载完整数据即可实现数据完整性验证。此外，贴合实际场景需求，近年一些方案进一步拓展功能边界，实现了DT-RDIC（Remote Data Integrity Checking with Data Ownership Transfer，支持所有权转移的远程数据完整性验证）[9-13]。例如企业因业务调整、合作变更需转移外包存储数据的所有权时，无需复杂操作即可完成合法权属转移与完整性同步验证，大幅提升了数据外包服务的灵活性与实用性。但现有方法仍存在以下几类显著局限性：

(1) 数据所有者的计算效率瓶颈。现有方案普遍存在大量的计算密集型密码学操作（如双线性映射和椭圆曲线乘法），给数据所有者带来了高昂的计算负担，在手机、物联网设备等资源受限的终端上这一问题尤为突出。

(2) 第三方审计机制的安全隐患。为降低用户负担，部分方法引入TPA执行完整性验证与所有权转移任务。然而，这类设计通常假设TPA完全可信，而实际环境中TPA可能因利益驱动或受外部胁迫，与云服务提供商合谋，造成审计结果失实与用户隐私泄露。

(3) 繁重的通信开销。多数现有方案对第三方验证的严重依赖，导致云服务提供商必须与之进行频繁交互，产生了显著的额外网络通信开销，构成系统性能瓶颈。

(4) 不可否认性机制的缺失。现有机制普遍缺乏有效的不可否认性支持。当数据所有者行为不端时——例如原所有者上传无效的同态可验证标签，或所有权转移中原所有者与现所有者共谋以伪造验证失败——CSP难以收集到不容抵赖的实质性证据进行追责。

针对上述挑战，本文提出一种基于TEE（Trusted Execution Environment，可信执行环境）的DT-RDIC方法（以下简称为本方案），主要贡献如下：

(1) 本方案将所有涉及完整性验证与所有权转移的关键操作置于TEE内部执行。借助TEE提供的硬件级隔离能力，本方案彻底摆脱了对TPA

的依赖。此外，通过远程认证机制，确保了数据所有者与TEE在完整性验证及所有权转移过程中通信的可靠性。

(2) 本文基于除TEE的隔离保护外所有实体均不可信的强威胁模型对本方案进行了安全性分析。结果表明，本方案能够有效抵御在该模型下可能出现的各类安全威胁。

(3) 理论分析表明，本方案不仅显著降低了数据所有者的计算开销，同时完全免除了云服务提供商与第三方审计机构之间的通信开销。本文基于Intel SGX（Intel Software Guard Extensions，英特尔软件保护扩展）[14]实现了原型系统。实验数据显示，本方案在计算效率上优于对比的RDIC与DT-RDIC方案，计算耗时降低幅度在10.1%至86.4%之间。

1　可信执行环境(TEE)简介

TEE是现代处理器架构中集成的硬件级安全隔离区域，作为设备计算环境的“安全可信子集”，专门承载高安全敏感度的代码执行与敏感数据处理任务。它在硬件层面为代码与数据提供机密性和完整性保护，可抵御操作系统、恶意软件等潜在威胁。TEE通过严格硬件隔离机制与受控交互流程实现安全防护，与REE（Rich Execution Environment，普通执行环境）在物理或逻辑层面完全隔离，仅能通过标准化安全接口（如Intel SGX的ECALL/OCALL、ARM TrustZone的SMC[15]）进行数据交互，且跨环境通信需经权限校验与数据加密，从根源阻断恶意程序非法访问。

TEE的主要特性包括：

(1) 数据隔离：TEE内部数据无法被未授权的其他应用访问。

(2) 计算隔离：TEE内部的计算过程无法被外部观察。

(3) 硬件信任根：依赖硬件支持（如CPU中的特定安全扩展）生成信任根，提供强安全保障。

(4) 可验证性：通过RA（Remote Attestation，远程认证）[16]，可在不可信的网络环境和硬件平台中创建TEE实例，并对其运行状态进行密码学验证。即使CSP具备基础设施管理权限，也无法突破TEE的隔离保护访问其内部数据与运行状态，从而在本质上提升了不可信环境下的数据安全保障能力。

目前TEE的主流实现方案包括Intel SGX、ARM TrustZone[17]、AMD SEV[18]等。

2　系统模型和威胁模型

2.1　系统模型

本方案引入了五个实体：CSP、PO（Previous Owner，原数据所有者）、CO（Current Owner，现数据所有者），以及PO与CO在CSP上通过RA创建的两个TEE（分别为TEE-PO与TEE-CO）。

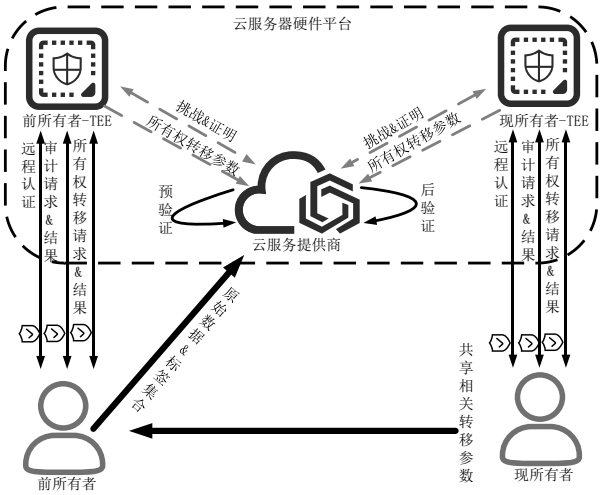


图1　系统架构图

如图1所示，本方案的主要流程如下：

(1) 系统初始化：PO、CSP与CO分别生成各自的公私钥对。随后，PO与CO生成必要的密钥与系统参数，并通过RA-TLS协议安全地传输至

其各自的TEE中。本阶段对应于系统初始化算法。

(2) 数据与HVT上传及预验证：PO首先生成数据的HVT，随后将数据块及其对应的HVT上传至CSP。CSP接收数据块与HVT后执行预验证，以确保其真实性和完整性。本阶段对应于HVT生成算法和预验证算法。

(3) 完整性验证：PO或已获得所有权的CO可发起验证流程，将验证请求发送至其对应的TEE（分别为TEE-PO或TEE-CO）。该TEE据此生成挑战并发送至CSP。随后CSP根据挑战中的随机密钥生成挑战集合并使用存储的数据块与HVT生成数据持有性证明，然后返回至发起挑战的TEE。TEE完成证明验证后，将验证结果返回给CSP和对应数据所有者。本阶段对应于挑战、证明生成与证明验证算法。

(4) 所有权转移及后验证：CO首先将必要参数安全共享给PO。随后，PO与CO分别向其所属TEE提交所有权转移请求。TEE-PO与TEE-CO将关键参数转发至CSP以触发所有权转移。所有权转移完成后，CSP对转移后的HVT执行后验证算法，以确保所有权转移的正确性与完整性。本阶段对应于HVT转移和后验证算法。

2.2 威胁模型

在当前主流远程数据完整性验证方案中，数据所有者通常被预设为可信实体，而TPA则遵循“诚实但好奇”的模型。与之相对，本方案采用更强的威胁假设：除通过RA-TLS建立的安全信道及TEE的隔离保护外，系统将所有其他参与实体视为潜在不可信对象。具体而言：

(1) 在验证阶段，CSP可能通过伪造证据掩盖数据损坏事实；在所有权转移阶段，CSP可能通过分析已知数据提取用户隐私信息，为后续伪造持有性证明创造条件。

(2) 在上传阶段，PO可能提交无效的HVT；在验证阶段，PO或CO可能操纵TEE伪造验证失败结果；在所有权转移阶段，PO和CO可能通过共谋使HVT失效，从而构陷CSP数据管理失职，达到非法索赔或商业诋毁的目的。

3 方案构造

本方案的受到文献[3]的启发，包含八个多项式时间算法：

(1) 初始化：$Setup(1^{\lambda})\rightarrow(params, sk_s, sk_c, sk_{sign}, sk_c^*, sk_{sign}^*)$：系统生成双线性映射参数：$G_1\times G_2\rightarrow G_T$，其中群$G_1$、$G_2$与$G_T$的阶为大素数$p$，并选取生成元$g_1\in G_1$与$g_2\in G_2$。设定密码学哈希函数$H$: $\{0,1\}^*\rightarrow G_1$，同时确定每个数据块的扇区数s。系统随机选取s个私密系数$r_1,\ldots,r_s\in Z_p$（由PO秘密保存），并计算相应的公开系数$u_j=g_1^{r_j}$，其中j=1, …, s。CSP生成其签名公私钥对(sk_s, pk_s)；PO生成其验证公私钥对$(sk_c\in Z_p, pk_c=g_2^{sk_c}\in G_2)$及签名公私钥对$(sk_{sign}, pk_{sign})$；CO则生成验证公私钥对$(sk_c^*\in Z_p, pk_c^*=g_2^{sk_c^*}\in G_2)$及签名公私钥对$(sk_{sign}^*, pk_{sign}^*)$。CSP本地保存其签名私钥$sk_s$；PO本地保存私钥$(sk_c, sk_{sign})$，并通过RA-TLS传输并保存到TEE-PO；CO本地保存私钥(sk_c^*, sk_{sign}^*)，并通过RA-TLS传输并保存到TEE-CO。公开参数$params=(e, p, G_1, G_2, G_T, g_1, g_2, H, s, \{u_1, ..., u_s\}, pk_s, pk_c, pk_{sign}^*, pk_{sign}, pk_c^*)$。在实际部署中，$pk_s$，$pk_c$，$pk_{sign}^*$，$pk_{sign}$和$pk_c^*$按照PKI规定的生命周期产生、管理、存储、分发和撤销。如需更改验证密钥（即(sk_c, pk_c)或(sk_c^*, pk_c^*)），除了生成新的私钥并申请新的公钥证书之外，PO/CO和CSP必须使用新的密钥运行后文所述的*HVTTran*和*PostCheck*算法，以更新所有的HVT。

(2) HVT生成：$HVTGen(params, \{r_1, \cdots, r_s\}, sk_c, sk_{sign}, name, F)\rightarrow(S, Sig(sk_{sign}, (name, F, S)))$：PO将文件$F$分割为$n$个数据块$(m_1, ..., m_n)$，每个块$m_i$进一步划分为$s$个扇区：$m_i=(m_{i_1}, \cdots, m_{is})$，其

中 $m_{ij} \in Z_p$。对每个数据块 $i \in \{1, \ldots, n\}$，计算其 HVT：$\sigma_i = (H(name \| i) \cdot g_1^{\sum_{j=1}^{s} r_j m_{ij}})^{sk_c}$，并构造 HVT 集合 $S = \{\sigma_1, \ldots, \sigma_n\}$，使用 PO 的签名私钥 sk_{sign} 生成 $Sig(sk_{sign}, (name, F, S))$，并将 $(name, F, S, Sig(sk_{sign}, (name, F, S)))$ 上传至 CSP。

(3) 预验证：$PreCheck(params, sk_c, name, F, S, Sig(sk_{sign}, (name, F, S))) \rightarrow (R', Sig(sk_s, (R', name)))$：CSP 将 $(name, F, S)$ 保存在临时空间，使用 pk_{sign} 验证签名 $Sig(sk_{sign}, (name, F, S))$ 的有效性。若验证失败，CSP 拒绝存储，删除 $(name, F, S)$ 并要求重传。若签名有效，CSP 随机生成临时密钥 tk'，并构造挑战集合 $Q' = \{(i, v_i) | i = 1, \ldots, n, v_i = f_{tk'}(name \| i) \in Z_p\}$。每次运行 *PreCheck* 时，$tk'$ 必须重新生成且只能使用一次，以避免重放攻击。CSP 计算证明 $P' = (\{\mu_1', \cdots, \mu_s'\}, \sigma')$，其中，$\mu_j' = \sum_{(i, v_i) \in Q'} v_i m, j = 1, ..., s, \sigma' = \prod_{(i, v_i) \in Q'} \sigma_i^{v_i}$。

CSP 执行核心验证，检查以下等式是否成立：

$$e(\sigma', g_2) \stackrel{?}{=} e((\prod_{(i, v_i) \in Q'} H(name \| i)^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j'}, pk_c)) \quad (1)$$

若等式成立，则设置验证结果 $R'=1$（通过），CSP 正式保存 $(name, F, S)$ 并返回签名 $Sig(sk_s, (1, name))$；否则设置 $R'=0$（不通过），CSP 拒绝存储，删除 $(name, F, S)$ 并返回签名 $Sig(sk_s, (0, name))$ 以要求重传。

(4) 挑战：$Challenge(params, sk_{sign}/sk_{sign}^*, name, c) \rightarrow chal$：PO 或已获得数据所有权的 CO 将 $(name, c)$ 通过 RA-TLS 安全传输至 TEE-PO/TEE-CO。TEE-PO/TEE-CO 随机生成临时密钥 $tk = (tk_1, tk_2)$，其中 tk_1 用于伪随机排列以选择被挑战的数据块索引，tk_2 用于伪随机函数以生成各挑战块对应的权重系数。每次运行 *Challenge* 时，tk 必须重新生成且只能使用一次，以避免重放攻击。c 为挑战块数量。设 t 为损坏块数和总块数 n 的比值，在 t=0.01，c=300 时可达到 95% 的检测率，在 t=0.01，c=460 时可达到 99% 的检测率[1]。用户可动态调整 c 以平衡可靠性与效率。此外，在后文所述的 *ProofGen* 算法中，CSP 会验证 $Sig(sk_{sign}/sk_{sign}^*, (tk, name, c))$ 以确保 c 未被篡改。TEE-PO/TEE-CO 生成 $Sig(sk_{sign}/sk_{sign}^*, (tk, name, c))$，并将 $chal = (tk, name, c, Sig(sk_{sign}/sk_{sign}^*, (tk, name, c)))$ 发送给 CSP。

(5) 证明生成：$ProofGen(params, Sig(sk_{sign}/sk_{sign}^*, (tk, name, c)), sk_s, F, S/S^*, chal) \rightarrow (P, Sig(sk_s, (tk, name, c, P)))$：CSP 验证 $Sig(sk_{sign}/sk_{sign}^*, (tk, name, c))$ 的有效性。若验证失败，则拒绝处理该挑战并返回签名 $Sig(sk_s, (tk, name, c))$ 以要求重传。若签名有效，CSP 使用临时密钥分量 tk_1 生成伪随机排列 $I = \pi_{tk_1}(n, c)$，其中 I 表示 c 个被挑战数据块索引的集合，并构造挑战集合 $Q = \{(i, v_i) | i = 1, \ldots, n, v_i = f_{tk_2}(name \| i)\}$。CSP 计算证明 $P = (\{\mu_1, ..., \mu_s\}, \sigma)$，其中，$\mu_j = \sum_{(i, v_i) \in Q} v_i m, j = 1, ..., s, \sigma = \prod_{(i, v_i) \in Q} \sigma_i^{v_i}$。CSP 生成 $Sig(sk_s, (tk, name, c, P))$，并将 $(P, Sig(sk_s, (tk, name, c, P)))$ 发送至 TEE-PO/TEE-CO。

(6) 证明验证：$ProofCheck(params, sk_c, sk_{sign}/sk_{sign}^*, tk, name, c, P, Sig(sk_s, (tk, name, c, P))) \rightarrow (R, Sig(sk_{sign}/sk_{sign}^*, (R, tk, name, c, P)))$：TEE-PO/TEE-CO 验证 $Sig(sk_s, (tk, name, c, P))$。若验证失败，则拒绝进行完整性验证并返回签名 $Sig(sk_{sign}/sk_{sign}^*, (tk, name, c, P))$ 以要求重传。否则，TEE-PO/TEE-CO 使用 tk_1 生成伪随机排列 $I = \pi_{tk_1}(n, c)$，并使用 tk_2 为每个 $i \in I$ 计算挑战权重 $v_i = f_{tk_2}(name \| i)$，从而重构挑战集合 Q。TEE-PO/TEE-CO 执行核心验证，检查以下等式是否成立：

$$e(\sigma, g_2) \stackrel{?}{=} e((\prod_{(i, v_i) \in Q} H(name \| i)^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, pk_c)) \quad (2)$$

若等式成立，则设置验证结果 $R=1$（通过），

否则R=0（不通过）。TEE-PO/TEE-CO生成$Sig(sk_{sign}/sk_{sign}^{*}, (R, tk, name, c, P))$，并将$(R, Sig(sk_{sign}/sk_{sign}^{*}, (R, tk, name, c, P)))$发送给CSP和PO/CO。

(7) 所有权转移：$HVTTran(params, sk_c, sk_c^{*}, name, n, F, S)\rightarrow S^{*}$：CO指定目标文件名$name$，随机选取$a\in\{0,1\}^{k}$，生成加密消息$Enc(pk_{sign}, name\|a)$及其签名$Sig(sk_{sign}^{*}, Enc(pk_{sign}, name\|a))$，并发送至PO（不使用RA-TLS），同时将$(name, a)$安全传输至TEE-CO（使用RA-TLS）。TEE-CO为每个$i\in\{1,\ldots,n\}$计算：$t_i^{*}=sk_c^{*}\cdot f_a(name\|i)$，生成签名$Sig(sk_{sign}^{*}, \{t_i^{*}\})$，并将二者发送至CSP。PO验证$Sig(sk_{sign}^{*}, Enc(pk_{sign}, name\|a))$。验证通过后，PO解密$Enc(pk_{sign}, name\|a)$获得$(name, a)$，并通过RA-TLS将其转发至TEE-PO。TEE-PO为每个$i\in\{1,\ldots,n\}$计算：$t_i=sk_c^{-1}\cdot(f_a(name\|i))^{-1}$，生成签名$Sig(sk_{sign}, \{t_i\})$，并将二者发送至CSP。CSP验证签名$Sig(sk_{sign}^{*}, \{t_i^{*}\})$和$Sig(sk_{sign}, \{t_i\})$。验证通过后，为每个数据块计算新HVT：

$$\sigma_i^{*}=\sigma_i^{t_i t_i^{*}} \tag{3}$$

构成转移后的HVT集合S^{*}，并保存在临时空间。

(8) 后验证：$PostCheck(params, sk_s, S^{*}, name)\rightarrow(R'^{*}, Sig(sk_s, (name, R^{*}')))$：CSP随机生成临时密钥$tk^{*}'$，并生成挑战集合$Q^{*}'=\{(i, v_i)|v_i=f_{tk^{*}}(name\|i)\in Z_p\}$。每次运行$PostCheck$时，$tk^{*}'$必须重新生成且只能使用一次，以避免重放攻击。CSP计算完整性证明$P^{*}'=(\{\mu_1^{*}', \cdots, \mu_s^{*}'\}, \sigma^{*}')$，其中：$\mu_j^{*}'=\sum_{(i,v_i)\in Q^{*}'} v_i m_{ij}, j=1,\ldots,s, \sigma^{*}'=\prod_{(i,v_i)\in Q^{*}'}\sigma_i^{*v_i}$。CSP执行核心验证，检查以下等式是否成立：

$$e(\sigma^{*}', g_2)\stackrel{?}{=}e(\prod_{(i,v_i)\in Q^{*}'}H(name\ \|\ i)^{v_i}, \prod_{j=1}^{s}u_j^{\mu_j^{*}'}, pk_c^{*}) \tag{4}$$

若等式(13)成立，则设置验证结果R^{*}'=1（成功），CSP确认所有权转移完成，使用S^{*}替换S并返回$Sig(sk_s, (name, R^{*}'))$，否则设置$R^{*}'$=0（失败），删除$S^{*}$，并等待新的$HVTTran$请求。

此外，为了避免过于繁琐，本章节的算法描述的省略了部分异常处理过程。对于可能发生的异常情况，首先，本方案所有的网络通信均采用TCP/IP作为运输层协议，自然实现了确认应答和超时重传机制以处理连接异常；其次，在各个算法中，本方案使用数字签名对传输的信息进行额外的身份识别和完整性验证，数字签名不正确也会触发重传机制。在收到正确的数字签名之前，任何请求均不可执行；最后，本方案中对数据的增加、修改和删除总是在CSP和其他实体的全部交互结束，且验证了签名的合法性和请求的正确性以后才能够执行，从而避免了因其他实体掉线或故障而破坏数据的一致性。因此，本方案能够处理各种非人为或人为原因导致的异常情况，从而避免因此造成的数据损坏。

4 正确性分析

根据$HVTGen$，对于每个数据块$m_i=(m_{i1}, \cdots, m_{is})$，其HVT为：$\sigma_i=(H(name\|i)\cdot g_1^{\sum_{j=1}^{s}r_j m_{ij}})^{sk_c}$，在$ProofGen$中，CSP生成的证明为$P=(\{\mu_1, \cdots, \mu_s\}, \sigma)$，其中$\mu_j=\sum_{(i,v_i)\in Q}v_i m, j=1,\ldots,s, \sigma=\prod_{(i,v_i)\in Q}\sigma_i^{v_i}$。因此，对于$ProofCheck$，我们有：

$$e(\sigma,g_2)=e(\prod_{(i,v_i)\in Q}\sigma_i^{v_i},g_2)=e(\prod_{(i,v_i)\in Q}\sigma_i^{v_i},g_2)=e(\prod_{(i,v_i)\in Q}((H(name||i)\cdot g_1^{\sum_{j=1}^{s}r_jm_{ij}})^{sk_c})^{v_i},g_2)$$
$$=e(\prod_{(i,v_i)\in Q}H(name||i)^{v_i}\cdot\prod_{(i,v_i)\in Q}g_1^{sk_cv_i\sum_{j=1}^{s}r_jm_{ij}},g_2)=e(\prod_{(i,v_i)\in Q}H(name||i)^{v_i}\cdot g_1^{sk_c\sum_{j=1}^{s}r_j\sum_{(i,v_i)\in Q}v_im_{ij}},g_2) \quad (5)$$
$$=e(\prod_{(i,v_i)\in Q}H(name||i)^{v_i}\cdot\prod_{j=1}^{s}g_1^{sk_cr_j\mu_j},g_2)=e(\prod_{(i,v_i)\in Q}H(name||i)^{v_i}\cdot\prod_{j=1}^{s}u_1^{\mu_j},g_2^{sk_c})$$
$$=e((\prod_{(i,v_i)\in Q}H(name\ \|\ i)^{v_i}\cdot\prod_{j=1}^{s}u_j^{\mu_j},pk_c))$$

因此，等式(2)是正确的，从而 *HVTGen*、*ProofGen* 和 *ProofCheck* 都是正确的。类似地，*PreCheck* 和 *PostCheck* 也都是正确的，因为其基本原理与 *ProofCheck* 相同。

对于 *HVTTran* 中的等式(3)，我们有：

$$\sigma_i^*=\sigma_i^{t_it_i^*}=((H(name||i)\cdot g_1^{\sum_{j=1}^{s}r_jm_{ij}})^{sk_c})^{sk_c^{-1}\cdot(f_a(name||i))^{-1}\cdot sk_c^*\cdot f_a(name||i)}=(H(name||i)\cdot g_1^{\sum_{j=1}^{s}r_jm_{ij}})^{sk_c^*} \quad (6)$$

可见新 HVT 具有与原始 HVT 相同的结构，只是将 PO 的私钥 sk_c 替换为 CO 的私钥 sk_c^*，因此 *HVTTran* 也是正确的。

5　安全性分析

(1) 可靠性：首先，我们证明 CSP 无法以不可忽略的概率伪造一个能够通过 TEE-PO 或 TEE-CO 验证的证明。

定理 1（不可伪造性）：在 CDH（Computational Diffie-Hellman，计算性迪菲-赫尔曼）问题困难性假设下，任何概率多项式时间的敌手 $\mathcal{A}$（恶意 CSP）在以下游戏中获胜的概率是可忽略的：

□ 建立：挑战者 $\mathcal{C}$ 运行 *Setup* 算法，将 pk_c 以及公开因子 $\{u_j\}$ 发送给敌手 $\mathcal{A}$，同时保密 sk_c 和 $\{r_j\}$。

□ 查询：$\mathcal{A}$ 可以自适应地提交数据块 $\overline{m_i}$、索引 i 和文件名 *name*，向 $\mathcal{C}$ 查询 $\overline{m_i}$ 对应的 HVT $\overline{\sigma_i}$（但不允许 $\mathcal{A}$ 使用相同的 i 和 *name* 查询不同的 $\overline{m_i}$ 的 HVT）。对于经过 *HVTGen* 查询的 $\overline{m_i}$ 的集合，$\mathcal{A}$ 可以请求执行 *Challenge*、*ProofGen* 和 *ProofCheck*，$\mathcal{C}$ 诚实响应。

□ 伪造：$\mathcal{A}$ 选择一个被查询过的文件 F（或其部分），$\mathcal{C}$ 随机生成挑战 *chal* 并发送给 $\mathcal{A}$。$\mathcal{A}$ 生成一个证明 $P''=(\{\mu_1'',\cdots,\mu_s''\},\sigma'')$ 作为响应。

□ 裁决：设真实的 $P=(\{\mu_1,\cdots,\mu_s\},\sigma)$，如果 P'' 能通过 *ProofCheck*，但 $P''\neq P$，则 $\mathcal{A}$ 获胜。

证明：　假设存在一个敌手 $\mathcal{A}$ 够以不可忽略的概率 ε 赢得上述游戏，那么我们可以构造一个模拟器 $\mathcal{B}$ 利用 $\mathcal{A}$ 以不可忽略的概率解决 CDH 问题。$\mathcal{B}$ 接收一个 CDH 挑战 (g_1, g_1^a, g_2^b)，其目标是计算 g_1^{ab}。

□ 建立：$\mathcal{B}$ 设置 $pk_c=g_2^b$（即隐含 $sk_c=b$，但 $\mathcal{B}$ 并不知道 b）。随机选择 $x_j\in Z_p$，并设置 $\{u_j=(g_1^a)^{x_j}|j=1,\ldots,s\}$。这隐含了 $r_j=ax_j$。将 pk_c 发送给 $\mathcal{A}$。

□ 查询：对于数据块 $\overline{m_i}$，$\mathcal{B}$ 需要计算 $\overline{\sigma_i}=(H(name||i)\cdot g_1^{\sum_{r=1}^{s}r_j\overline{m_{ij}}})^b$。由于 $\mathcal{B}$ 不知道 b，它通过编程随机预言机 H 来模拟：随机选择 $h_i\in G_1$，并“定义” $H(name||i)=h_i\cdot g_1^{-a\sum_{r=1}^{s}x_j\overline{m_{ij}}}$。那么

$$\sigma_i=(h_i\cdot g_1^{-a\sum_{r=1}^{s}x_j\overline{m_{ij}}}\cdot g_1^{\sum_{r=1}^{s}r_j\overline{m_{ij}}})^b=(h_i\cdot g_1^{-\sum_{r=1}^{s}r_j\overline{m_{ij}}}\cdot g_1^{\sum_{r=1}^{s}r_j\overline{m_{ij}}})^b=h_i^b \quad (7)$$

□ 伪造：$\mathcal{A}$ 选择一个被查询过的文件 $F=(m_1,\ldots,m_n)$（或其部分），$\mathcal{B}$ 随机生成挑战 *chal* 并发送给 $\mathcal{A}$。$\mathcal{A}$ 生成一个证明 $P''=(\{\mu_1'',\cdots,\mu_s''\},\sigma'')$ 作为响应。

□ 裁决：如果 P'' 能通过 *ProofCheck*，则有：

$$e(\sigma'',g_2)=e(\prod_{(i,v_i)\in Q}H(name||i)^{v_i}\cdot\prod_{j=1}^{s}u_j^{\mu_j''},pk_c) \quad (8)$$

$\mathcal{B}$自己计算真实的$P=(\{\mu_1,\cdots,\mu_s\},\sigma)$，我们也有：

$$e(\sigma,g_2)=e(\prod_{(i,v_i)\in Q}H(name||i)^{v_i}\cdot\prod_{j=1}^{s}u_j^{\mu_j},pk_c) \tag{9}$$

设$\{\Delta\mu_j=\mu_j''-\mu_j\}$，显然，若$P''\neq P$，则至少存在一个$\Delta\mu_j\neq 0$（否则$\sigma''=\sigma$，从而$P''=P$）。(8)和(9)两式相除，我们得到：

$$e(\sigma''\sigma^{-1},g_2)=e(\prod_{j=1}^{s}u_j^{\Delta\mu_j},pk_c)=$$

$$e(\prod_{j=1}^{s}(g_1^{ax_j})^{\Delta\mu_j},g_2^{b})=e((g_1^{ab})^{\sum_{j=1}^{s}x_j\Delta\mu_j},g_2) \tag{10}$$

因为$\{v_i^{(j)}=f_{tk_2^{(j)}}(name||i)\mid i=1,\ldots,c,j=1,\ldots,s\}$，$f$是一个安全的伪随机函数，因此任何一个线性方程组的系数矩阵均可视为随机的，即除了$1/p$的概率以外均为满秩矩阵。因此，$\mathcal{C}$可以通过解这s个线性方程组得到原始的$\{m_{ij}\}$，除了可忽略的概率以外。因此，本方案的*ProofCheck*是可靠的。而本方案的*PreCheck*和*PostCheck*的原理和*ProofCheck*相同，因此也是可靠的。

(2) 不可抵赖性：不可抵赖性要求任何参与方（特别是PO和CO）无法事后否认其已确认的操作结果，也无法在数据状态正确时成功诬陷CSP。本方案通过数字签名和HVT的不可伪造性为所有参与方提供了不可否认的证据。

□ 针对恶意PO/CO的不可抵赖性：首先，*PreCheck*和*PostCheck*已经阻止了PO/CO故意上传或生成错误的F和S/S^*。当PO/CO试图在*ProofCheck*时诬陷CSP丢失数据时，CSP可以出示以下证据：

● 有效的$Sig(sk_{sign}/sk_{sign}^*,(tk,name,c))$，证明当前的挑战确实来源于PO/CO。

从而$(g_1^{ab})^{\sum_{j=1}^{s}x_j\Delta\mu_j}=\sigma''\sigma^{-1}$。除非$\sum_{j=1}^{s}x_j\Delta\mu_j=0$（但对于随机的$\{x_j\}$和不全为零的$\{\Delta\mu_j\}$，此式成立的概率仅为$1/p$，是可忽略的），我们得到$g_1^{ab}=(\sigma''\sigma^{-1})^{(\sum_{j=1}^{s}x_j\Delta\mu_j)^{-1}}$，这与CDH问题困难性假设矛盾。定理1证毕。

其次，在$\{\mu_j\}$不可伪造的前提下（定理1），对于一个特定的文件F（或其部分），$\mathcal{B}$可以每次选择不同的tk_2，重复运行以上游戏c次，得到s个线性方程组：

$$\begin{cases}\mu_1^{(1)}=\sum_{(i,v_i^{(1)})\in Q^{(1)}}v_i^{(1)}m_{i1}\\ \cdots\\ \mu_1^{(c)}=\sum_{(i,v_i^{(c)})\in Q^{(c)}}v_i^{(c)}m_{i1}\end{cases},\ldots,\begin{cases}\mu_s^{(1)}=\sum_{(i,v_i^{(1)})\in Q^{(1)}}v_i^{(1)}m_{is}\\ \cdots\\ \mu_s^{(c)}=\sum_{(i,v_i^{(c)})\in Q^{(c)}}v_i^{(c)}m_{is}\end{cases} \tag{11}$$

● 使用F、S/S^*、tk、$name$和c重新计算P。

● 自己使用pkc运行*ProofCheck*得到R=1。

● 有效的$Sig(sk_s,(tk,name,c,P))$和$Sig(sk_{sign}/sk_{sign}^*,(R=0,tk,name,c,P))$。

如果P可由原始的F、S/S^*计算得到，且*ProofCheck*得到R=1，则同时说明了挑战的数据块和HVT均来自于PO/CO和原始的F、S/S^*。但TEE-PO/TEE-CO却声称验证失败（R=0），只能说明TEE-PO/TEE-CO在*ProofCheck*时返回了错误的验证结果。

□ 针对恶意CSP的不可抵赖性：定理1已经证明了持有性证明P的不可伪造性。PO/CO也持有CSP的签名$Sig(sk_s,(tk,name,c,P))$，防止CSP在生成P后抵赖。

(3) 隐私保护性：隐私保护性要求CSP不能在*HVTTran*中通过分析已知数据提取用户隐私信息。在*HVTTran*中CSP获得的信息是$\{t_i^*=sk_c^*\cdot f_a(name||i)|i=1,\ldots,n\}$和$\{t_i=sk_c^{-1}\cdot(f_a(name||i))^{-1}|i=1,\ldots,n\}$，其中$sk_c^*$和$sk_c$属于隐私信息。我们将其改写为$\{sk_c^*=t_i^*\cdot(f_a(name||i))^{-1}|i=1,\ldots,n\}$和$\{sk_c=$

$t_i^{-1}\cdot(f_a(name||i))^{-1}|i=1,\ldots,n\}$。因为$f$是一个安全的伪随机函数，因此在不知道$a$的情况下，$sk_c^*$可以视为以$(f_a(name||i))^{-1}$为密钥对$t_i^*$进行OTP（One Time Pad，一次一密）加密，sk_c可以视为以$(f_a(name||i))^{-1}$为密钥对t_i^{-1}进行OTP加密。此加密是信息论安全的，即CSP猜中sk_c^*和sk_c的真实值的概率仅为$1/p$。

6　性能分析

本节对本方案进行性能评估与实验分析。我们选取了三种具有代表性的数据完整性验证方案与本方案进行对比，包括两种支持所有权转移的方案（文献[9]、文献[12]）和一种不支持所有权转移的方案（文献[7]）。

6.1　理论分析

表1展示了各阶段计算开销的比较结果，其中$|F|$表示原始数据大小，$|f|$表示被挑战数据大小，L表示最小数据单元大小（对于未划分扇区的方案，如文献[12]及文献[7]，最小数据单元为数据块；对于划分扇区方案，如文献[9]及本方案，最小数据单元为扇区）。设s为每个数据块的扇区数，H、M、E、P和I分别表示单次哈希运算、大整数乘法、椭圆曲线乘法、双线性映射和逆运算的开销；加法运算开销可忽略不计。虽然本方案引入了数字签名与加密操作，但其仅产生$O(1)$级别开销，故同样可忽略。

分析结果表明，在HVT生成阶段，本方案的开销最低。本方案的数据块数量$n=\frac{|F|}{Ls}$显著少于未划分扇区的方案（文献[12]与文献[7]），使得总计算量远低于文献[12]与文献[7]（其中$\frac{|F|}{L}$远大于$\frac{|F|}{Ls}$）；在证明生成阶段，文献[9]的开销略低于本方案，但两者的性能差距很小；在证明验证阶段，本方案的开销最低，文献[9]次之，而文献[7]由于配对运算次数与挑战块数$\frac{|f|}{L}$成正比，在$\frac{|f|}{L}$较大时将产生显著计算开销；在所有权转移阶段，本方案的开销最低，而其他对比方案均采用更高频次的高强度运算，开销较高。

表2展示了各阶段通信开销的比较结果，其中$|F|$、$|f|$和L的含义与前述保持一致。本方案与所有对比方案均基于椭圆曲线与双线性对构造，若采用BN254曲线及双线性对$G_1\times G_2\to G_T$，则表中I、$|Z_p|$、$|G_1|$、$|G_2|$、$|G_T|$与H分别表示小整数（64比特长整型）、Z_p元素（256比特）、G_1元素（256比特，压缩点）、G_2元素（512比特，压缩点）、G_T元素（3072比特）及哈希值（256比特，

表1　计算开销对比表

方案	HVT生成	证明生成	证明验证	所有权转移
文献[12]	$\frac{\|F\|}{L}H+\frac{2\|F\|}{L}M+\frac{2\|F\|}{L}E$	$(\frac{2\|f\|}{L}-1)M+\frac{\|f\|}{L}E$	$3P+(\frac{\|f\|}{L}+1)H+(\frac{\|f\|}{L}+1)M+(\frac{\|f\|}{L}+2)E$	$4H+(\frac{2\|F\|}{L}+9)M+(\frac{5\|F\|}{L}+2)E+7I$
文献[9]	$(\frac{\|F\|}{Ls}+1)H+\frac{\|F\|}{L}M+(\frac{\|F\|}{L}+\frac{\|F\|}{Ls}+1)E$	$(\frac{2\|f\|}{Ls}-1)M+\frac{\|f\|}{Ls}E$	$4P+(\frac{\|f\|}{Ls}+1)H+\frac{\|f\|}{Ls}M$	$(\frac{2\|F\|}{Ls}+1)H+\frac{2\|F\|}{Ls}M+(\frac{\|F\|}{Ls}+3)E+\frac{\|F\|}{Ls}I$
文献[7]	$\frac{\|F\|}{L}H+\frac{\|F\|}{L}M+(\frac{2\|F\|}{L}+1)E$	$H+P+\frac{2\|f\|}{L}M+(\frac{\|f\|}{L}+1)E$	$(\frac{\|f\|}{L}+2)H+\frac{\|f\|}{L}P+(\frac{\|f\|}{L}-1)M+(\frac{\|f\|}{L}+1)E$	N/A
本方案	$\frac{\|F\|}{Ls}H+\frac{\|F\|}{Ls}(s+1)E$	$\frac{\|f\|}{L}M+\frac{\|f\|}{Ls}E$	$\frac{\|f\|}{Ls}H+(\frac{\|f\|}{Ls}+s)E+2P$	$\frac{3\|F\|}{Ls}M+\frac{\|F\|}{Ls}I+\frac{\|F\|}{Ls}E$

SHA-256）的尺寸，此外，本方案还采用了ECDSA签名与ECIES加密机制，因此|*Sig*|=512比特，|*Enc*|=1024比特。

在HVT生成阶段，文献[9]的开销最低，但本方案与文献[9]相比仅相差一个固定大小的数字签名，几乎是可忽略的；在挑战阶段，本方案的开销最低。本方案的通信仅发生于数据所有者与其TEE之间，传输内容仅包含文件名*name*与挑战块数*c*，因此通信开销恒定为*H*+*I*；在证明生成阶段，由于证明验证由部署于CSP的TEE-PO执行，其与CSP之间的数据传输属于内部通信，因此该阶段的对外通信开销可视为0；在所有权转移阶段，文献[12]的开销最低，为恒定值约1920比特，但本方案也实现了恒定开销，约为2560比特，与文献[12]差距很小。

6.2 实验结果

本方案原型系统采用C++语言开发，采用Gramine（轻量级隔离运行时，用于将原生Linux应用程序快速迁移至Intel SGX可信执行环境）、mcl（高效椭圆曲线与双线性对运算库）以及OpenSSL（通用密码学算法库）实现主要功能。CSP节点部署于配备Intel Xeon E-2378G@2.80 GHz处理器的Linux服务器，该CPU支持Intel SGX，支持最大128 TB的飞地空间，系统内存为128 GB。PO与CO部署于另一台硬件配置相同但不启用Intel SGX的Linux服务器，仅需安装RA-TLS相关的运行库。为更客观地评估算法改进的效果，我们额外实现了本方案的非TEE版本（使用相同的算法库和C++代码，但禁用Gramine的任何安全隔离功能），以证明本方案的性能提升源自算法的改进而非TEE本身。对比方案的仿真实现同样基于C++，mcl和OpenSSL，并在相同的硬件平台上执行（但未启用Intel SGX功能）。

为保障对比实验的公平性，鉴于部分方案（如文献[9]与本方案）采用数据块划分扇区的结构，而其他方案（如文献[12]与文献[7]）未采用此结构，我们将性能对比划分为两组进行。在与文献[12]及文献[7]的对比中，本方案设定扇区数*s*=1（等效于无扇区划分），所有方案数据块大小统一设为16字节。在与文献[9]的对比中，各方案统一设定*s*=32，扇区大小为16字节。实验结果表明，尽管TEE不会带来效率的优势，反而额外引入了6.4%-15.0%的计算开销（主要源自Enclave上下文切换和EPC分页[14]），本方案在各阶段的计算开销仍然低于对比方案，降低幅度在10.1%至86.4%之间。

在HVT生成阶段，图2(a)和图2(b)展示了数据块数量从500增至5000时各方案的计算开销。图2(a)显示，当数据块数量为5000时，文献[12]与文献[7]分别耗时715.2毫秒与761.2毫秒，本方案的非TEE版本耗时481.8毫秒，TEE版本耗时530.6毫秒（相比非TEE版本增加了10%的开销，但相比其他方案减少了25.8%与30.3%的开销）。图2(b)进一步表明，在相同数据规模下，文献[9]耗时为5826.5毫秒，本方案的非TEE版本耗时718.2毫秒，TEE版本耗时790.8毫秒（相比非

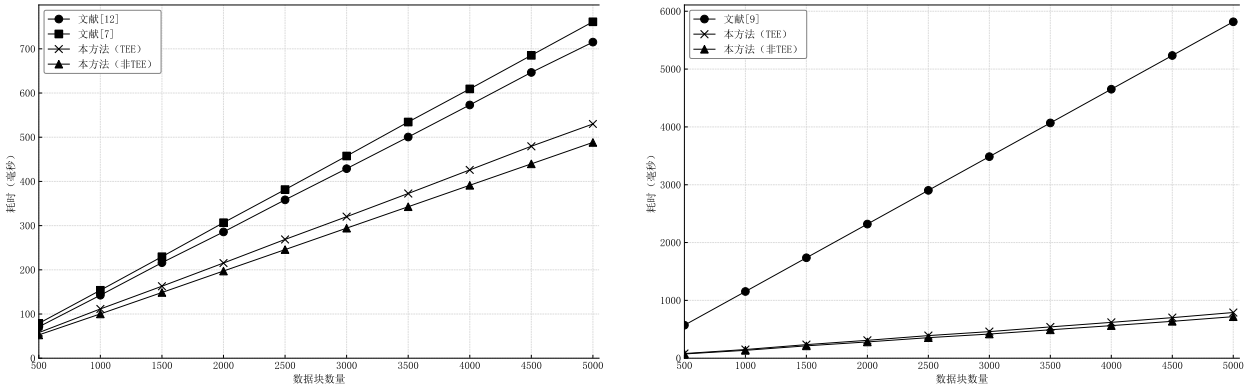
表2 通信开销对比表

方案	HVT生成	挑战	证明生成	所有权转移
文献[12]	$\frac{\lvert F\rvert}{L}\lvert G_1\rvert$	$\frac{\lvert f\rvert}{L}(I+\lvert Z_q\rvert)$	$\lvert Z_q\rvert+\lvert G_1\rvert$	$3\lvert G_1\rvert+6\lvert Z_q\rvert$
文献[9]	$\frac{\lvert F\rvert}{Ls}\lvert G_1\rvert$	$I+2\lvert Z_q\rvert$	$3\lvert G_1\rvert+s\lvert Z_q\rvert$	$\frac{\lvert F\rvert}{Ls}(\lvert Z_q\rvert+\lvert G_1\rvert)$
文献[7]	$(\frac{\lvert F\rvert}{L}+2)\lvert G_1\rvert+\lvert G_2\rvert$	$\frac{\lvert f\rvert}{L}\left(I+\lvert Z_q\rvert\right)+2\lvert G_2\rvert+2\lvert G_T\rvert+\lvert Z_q\rvert+H$	$2\lvert G_1\rvert+\lvert G_2\rvert+H$	N/A
本方案	$\frac{\lvert F\rvert}{Ls}\lvert G_1\rvert+\lvert Sig\rvert$	$H+I$	0	$\lvert Enc\rvert+\lvert Sig\rvert+2H$

TEE 版本增加了 10% 的开销，但相比其他方案减少了 86.4% 的开销）。

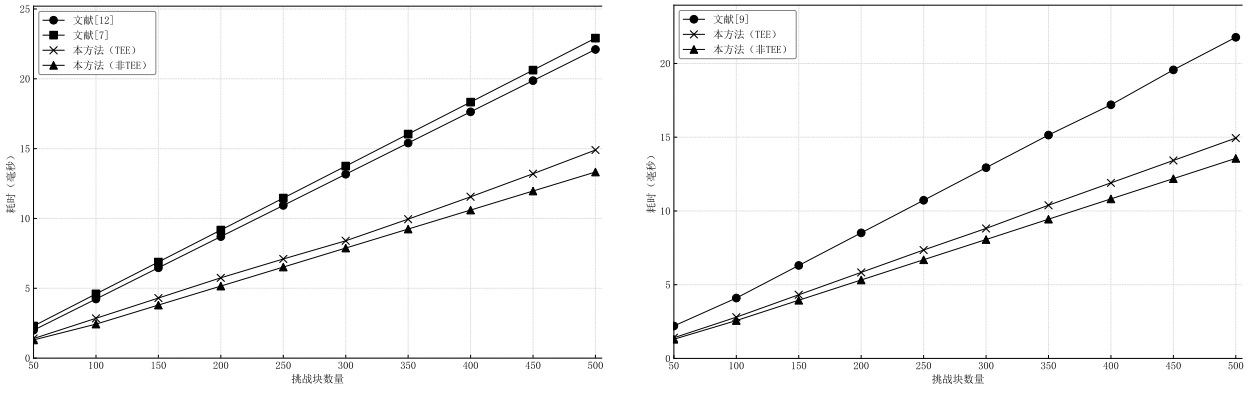
在证明生成阶段，图 3(a)和图 3(b)展示了挑战块数量从 50 增至 500 过程中各方案的计算开销变化。图 3(a)显示，当挑战块数量达到 500 时，文献[12]与文献[7]的计算开销分别为 22.1 毫秒与 23.4 毫秒，本方案的非 TEE 版本耗时 13.6 毫秒，TEE 版本耗时 15.1 毫秒（相比非 TEE 版本增加了 10% 的开销，但相比其他方案减少了 31.7% 与 35.5% 的开销）。图 3(b)表明，在相同挑战规模下，文献[9]的开销为 23.2 毫秒，本方案的非 TEE 版本耗时 15.1 毫秒，TEE 版本耗时 16.6 毫秒（相比非 TEE 版本增加了约 10% 开销，但相比其他方案减少了 28.4% 的开销）。

在证明验证阶段，图 4(a)和图 4(b)展示了挑战块数量从 50 增至 500 过程中各方案的计算开销。图 4(a)显示，当挑战块数量为 500 时，文献[12]与文献[7]方案分别耗时 62.4 毫秒与 65.1 毫秒，本方案的非 TEE 版本耗时 52.6 毫秒，TEE 版本耗时 56.1 毫秒（相比非 TEE 版本增加了 6.5% 的开销，但相比其他方案减少了 10.1% 与 13.8% 的开销）。图 4(b)进一步表明，在相同挑战规模下，文献[9]方案耗时 64.2 毫秒，本方案的非 TEE 版本耗时 41.3 毫秒，TEE 版本耗时 44.2 毫秒（相比非 TEE 版本增加了 6.5% 的开销，但相比其他方案减少了 31.2% 的开销）。



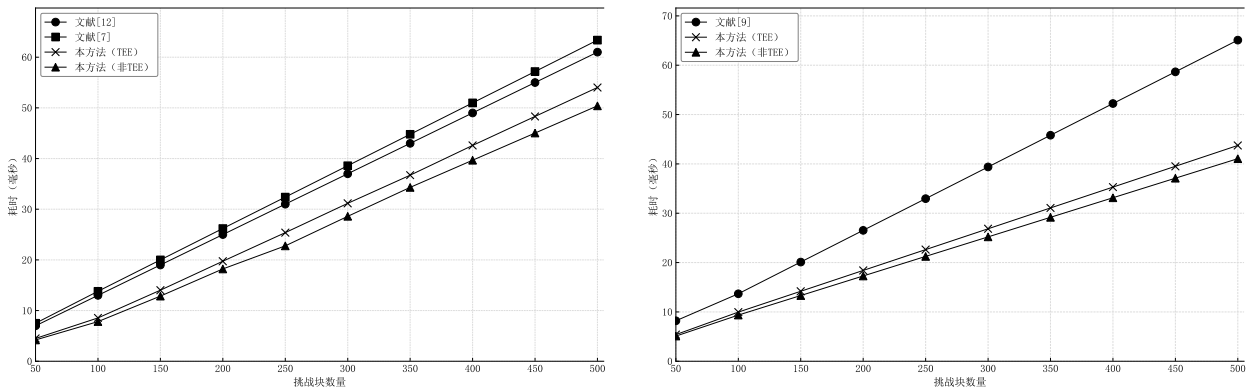
(a)　　(b)

图 2(a)　HVT 生成开销对比图（不划分扇区）(b) HVT 生成开销对比图（划分扇区）



(a)　　(b)

图 3(a)　证明生成开销对比图（不划分扇区）(b) 证明生成计算开销对比图（划分扇区）



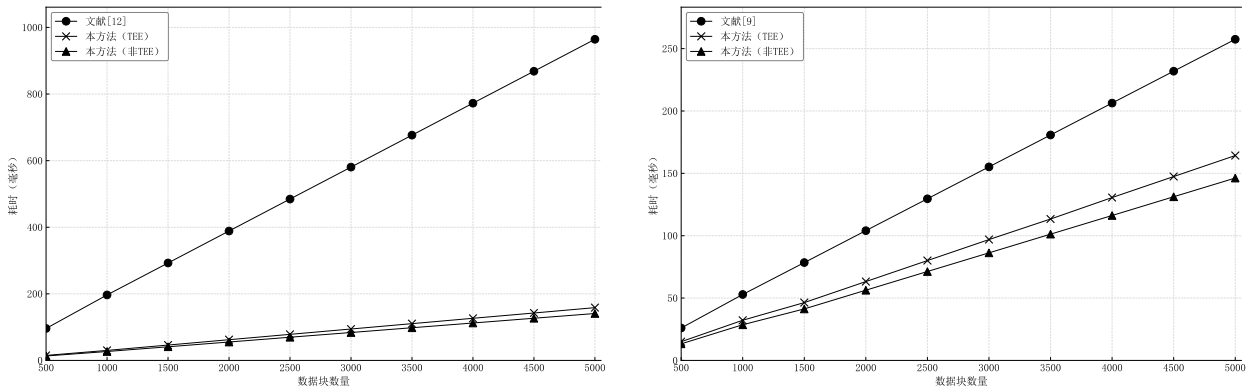
(a) (b)

图4(a) 证明验证开销对比图（不划分扇区）(b)证明验证开销对比图（划分扇区）

在HVT转移阶段，图5(a)和图5(b)展示了数据块数量从500增至5000时各方案的计算开销。图5(a)显示，当数据块数量为5000时，文献[12]需耗时949.1毫秒，本方案的非TEE版本耗时139.6毫秒，TEE版本耗时161.2毫秒（相比非TEE版本增加了12.5%的开销，但相比其他方案减少了83.0%的开销）。图5(b)进一步表明，在相同数据规模下，文献[9]需耗时257.1毫秒，本方案的非TEE版本耗时156.4毫秒，TEE版本耗时176.3毫秒（相比非TEE版本增加了12.5%的开销，但相比其他方案减少了31.4%的开销）。

结论

本文提出了一种基于可信执行环境的支持所有权转移的远程数据完整性验证方案，本方案不仅将数据所有者从繁重的计算任务中解放出来，也彻底摆脱了对传统第三方审计机构的依赖。本方案将完整性验证与所有权转移的关键操作置于TEE中执行，使数据所有者仅承担低开销的密码学操作，并完全消除了云服务提供商与外部审计者之间的通信开销。安全性分析表明，本方案在CDH困难性假设下具备可靠性，且通过伪随机函数和数字签名有效地防御包括证据伪造、密钥推断和共谋诬陷在内的多种威胁。本文基于Intel SGX实现的原型系统验证了方案的可行性，实验结果表明，本方案在HVT生成、所有权转移等阶段的性能优于对比的方案，计算开销降低幅度达



(a) (b)

图5(a) 所有权转移开销对比图（不划分扇区）(b) 所有权转移开销对比图（划分扇区）

10.1%–86.4%，展现出在移动终端与物联网等资源受限场景下的突出适用性。未来工作将致力于扩展方案的应用场景，例如动态更新、数据去重等。

参考文献：

[1] Ateniese G, Burns R, Curtmola R, et al. Provable Data Possession at Untrusted Stores[C]// Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, Virginia, USA: ACM, 2007: 598-609.

[2] Juels A, Kaliski Jr B S. PORs: Proofs of Retrievability for Large Files[C]// Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, Virginia, USA: ACM, 2007: 584-597.

[3] Shacham H, Waters B. Compact Proofs of Retrievability[C]// Shacham H, Waters B. Compact Proofs of Retrievability[C]// Advances in Cryptology - ASIACRYPT 2008. Heidelberg, Germany: Springer, 2008: 90-107.

[4] Wang Q, Wang C, Li J, et al. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing[C]// European Symposium on Research in Computer Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009: 355-370.

[5] Wang C, Chow S S M, Wang Q, et al. Privacy-preserving Public Auditing for Secure Cloud Storage[J]. IEEE transactions on Computers, 2011, 62(2): 362-375.

[6] Erway C C, Küpçü A, Papamanthou C, et al. Dynamic Provable Data Possession[J]. ACM Transactions on Information and System Security (TISSEC), 2015, 17(4): 1-29.

[7] Yu Y, Au M H, Ateniese G, et al. Identity-based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage[J]. IEEE Transactions on Information Forensics and Security, 2016, 12(4): 767-778.

[8] 杨帆, 袁艺林, 张邓凡, 等. 支持审计者更换和数据动态的云数据完整性审计方案[J]. 信息安全学报, 2025, 10(03): 197-208.
YANG Fan, YUAN Yilin, ZHANG Dengfan, et al. Cloud Data Integrity Audit Scheme that Supports Auditor Replacement and Data Dynamics[J]. Journal of Cyber Security, 2025, 10(03): 197-208.

[9] Wang H, He D, Fu A, et al. Provable Data Possession with Outsourced Data Transfer[J]. IEEE Transactions on Services Computing, 2019, 14(6): 1929-1939.

[10] Shen J, Guo F, Chen X, et al. Secure cloud auditing with efficient ownership transfer[C]//European Symposium on Research in Computer Security. Guildford, UK: Springer International Publishing, 2020: 611-631.

[11] Shen J, Chen X, Wei J, et al. Blockchain-based Accountable Auditing with Multi-ownership Transfer[J]. IEEE Transactions on Cloud Computing, 2022, 11(3): 2711-2724.

[12] Huang Y, Shen W, Qin J. Certificateless Cloud Storage Auditing Supporting Data Ownership Transfer[J]. Computers & Security, 2024, 139: 103738.

[13] 殷新春, 王经纬, 宁建廷. 支持高效数据所有权共享的动态云存储审计方案[J]. 软件学报, 2025, 36(07): 3306-3320.
YIN Xinchun, WANG Jingwei, NING Jianting. Dynamic Cloud Storage Auditing Scheme with Efficient Data Ownership Sharing[J]. Journal of Software, 2025,36(07): 3306-3320.

[14] McKeen F, Alexandrovich I, Anati I, et al. Intel® Software Guard Extensions (Intel® SGX) support for dynamic memory management inside an enclave[M]//Proceedings of the Hardware and Architectural Support for Security and Privacy 2016. 2016: 1-9.

[15] Ngabonziza B, Martin D, Bailey A, et al. TrustZone Explained: Architectural Features and Use Cases[C]// 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC). IEEE, 2016: 445-451.

[16] Gurevin D, Jin C, Nguyen PH, et al. Secure remote attestation with strong key isolation guarantees[J]. IEEE Transactions on Computers, 2023, 74(3): 848-859.

[17] Jang J, Choi C, Lee J, et al. Privatezone: Providing a private execution environment using arm trustzone[J]. IEEE Transactions on Dependable and Secure Computing, 2016, 15(5): 797-810.

[18] Zhao S, Li M, Zhangyz Y, et al. vsgx: Virtualizing sgx enclaves on amd sev[C]//2022 IEEE Symposium on Security and Privacy (SP). IEEE, 2022: 321-336.

彭溯（1983-），男，博士研究生，海南大学网络空间安全学院副教授，主要研究方向为云计算安全。

冯俐（2001-），男，海南大学网络空间安全学院硕士研究生在读，主要研究方向为云存储安全和可信执行环境。

尹贻然（2000-），男，海南大学网络空间安全学院硕士研究生在读，主要 研究方向为远程数据完整性验证与应用密码学。