

电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

博士学位论文

DOCTORAL DISSERTATION



论文题目 云数据完整性与可用性研究

学科专业 信息安全

学 号 201011060111

作者姓名 张新鹏

指导教师 许春香 教 授

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名: 张新鹏 日期: 2016年6月28日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后应遵守此规定)

作者签名: 张新鹏 导师签名: 汗森

日期: 2016年6月28日

分类号 _____ 密级 _____

UDC ^{注1} _____

学 位 论 文

云数据完整性与可用性研究

(题名和副题名)

张新鹏

(作者姓名)

指导教师

许春香

教 授

电子科技大学

成 都

(姓名、职称、单位名称)

申请学位级别 博士 学科专业 信息安全

提交论文日期 2016.04.15 论文答辩日期 2016.06.06

学位授予单位和日期 电子科技大学 2016 年 06 月 28 日

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号

Research on Cloud Data Integrity and Usability

**A Doctoral Dissertation Submitted to
University of Electronic Science and Technology of China**

Major: **Information Security**

Author: **Zhang Xinpeng**

Supervisor: **Prof. Xu Chunxiang**

School: **School of Computer Science & Engineering**

摘要

近年来，随着网络技术的飞速发展，个人、公司、组织产生了海量的数据。数据规模的增长速度远远超过了计算机处理和存储能力的增长速度。面对这样的现实和趋势，为了方便数据访问，不受制于本地数据存储和维护代价，用户不得不将海量数据存储在远程云服务器上。然而，数据的外包存储为保障用户数据的完整性与可用性带来了诸多挑战，例如不可避免的数据损坏、云服务提供商恶意篡改等完整性挑战，以及面向数据完整性验证的验证效率、用户可撤销、密钥更新、针对不同访问终端密文类型可修改等可用性挑战阻止了云存储服务的迅速推广应用。因此，本论文开展云数据完整性与可用性研究，所取得的研究成果主要体现在四个方面：

1. 高效安全的云存储数据完整性验证方案研究

(1) 提出了一个轻量级的云存储数据自审计方案。基于椭圆曲线数字签名算法(ECDSA)，构造了线性同态标签，使得数据所有者可以在不取回数据的前提下，检验数据的完整性，从而显著地减少了通信代价和计算开销。

(2) 设计了一个高效的云存储数据公开完整性验证方案。该方案利用变型的 Schnorr 签名算法与同态消息认证码技术，减少存储认证信息的存储空间、审计证明响应信息的计算开销以及验证者与云服务器之间的通信代价。并使用随机掩饰码技术确保第三方审计者不能通过审计信息恢复用户的原始数据块，有效的保护了用户的隐私。

文中对两个方案的安全性进行了严格证明，并对方案功能进行了扩展，使其能够分别支持数据动态操作与同时处理批量审计任务。提出的新方案由于未使用双线性对运算，因而效率更高，能够有效应用于分布式传感器网络等计算、存储和带宽受限的应用环境。

2. 支持用户可撤销的云存储数据完整性验证方案研究

(1) 利用聚合签名技术提出了一个支持用户可撤销的云存储数据公开完整性验证方案，有效地实现了当前用户在不取回数据的前提下对云服务器上所有历史用户的存储数据进行审计，满足了因当前用户离职而产生的数据移交需求。同时云用户可以委托第三方审计者执行安全审计任务。本文在随机预言机模型下证明了方案的安全性，并证明了该方案可以有效抵御撤销用户与恶意云服务的合谋攻击。

(2) 利用单向代理重签名技术提出了一个具有隐私保护功能的支持用户可撤

销的云存储数据公开完整性验证方案。该方案中的代理重签名密钥由当前用户私钥结合已撤销用户公钥生成，避免了私钥泄露问题，能够安全地实现数据所有权的转移。方案采用了随机掩饰码技术能有效防止好奇的第三方审计者恢复原始数据块。该方案同样在随机预言机模型下证明了安全性，可以抵制由于已撤销用户与云服务商合谋而产生的攻击。

3. 支持密钥更新的高效云存储数据公开完整性验证方案研究

设计了支持密钥更新功能的高效云存储数据公开完整性验证方案，该方案由零知识证明系统、代理重签名技术以及线性同态验证标签组合设计而成，有效地解决了密钥更新的问题。当云用户的密钥到期需要更新时，不再需要从云端下载他的数据。而是，从云端下载一个与整个文件相比非常小的认证符作为替代。这样用户可以有效地更改验证标签，从而大大地降低了通信和计算代价。方案同时被证明在随机预言机模型下是可证明安全的。最后提出了一个方案的实施原型，实验结果与理论分析相吻合，表明该方案适用于嵌入式云存储系统等计算、存储、带宽受限的应用环境。

4. 密文类型可修改的云存储数据可用性方案研究

此外在对云存储数据完整性研究的同时还对云存储数据的可用性进行了研究。针对不同访问终端密文类型可修改问题，首先指出了 Liu 等人所提方案的两个安全漏洞，即类型修改缺乏验证以及类型修改引起的新的条件性选择明文攻击问题。其次，进一步设计了一个云计算环境下支持类型可修改的基于身份代理重加密方案。该方案在具有传统代理重加密方案的核心功能的同时，密文的拥有者还可以随时修改密文的类型信息。与相关研究工作比较，该方案更加安全实用。

关键词：云存储，完整性验证，用户可撤销，密钥可更新，代理重加密，代理重签名

ABSTRACT

With the rapid development of network technology, a large amount of data have been gathered and produced by individuals, companies and organizations. The growth of the data scale is far more than the growth of the processing and storage capacity of computers. Faced with this reality and the trend, users have to outsource their data to the cloud such that they can make access to their data more conveniently and can be relieved from the burden of local data storage and maintenance cost. Although the cloud storage service makes these advantages more appealing than ever before, it also introduces new challenges towards the integrity, and availability of user's outsourced data. For example, integrity challenges of data corruption are inevitable, cloud service providers are dishonest. The research challenges about data usability towards data integrity such as auditing efficiency, user revocability, key update, according to the different access terminal ciphertext types can be modified etc. prevent the rapid popularization and application of cloud storage service. In this dissertation, we focus on the cloud data integrity and availability research, and the research outcomes are four folds:

1. Research on efficient and secure cloud data integrity verification schemes. (1) An efficient and light-weight data owner auditing scheme for secure cloud storage was proposed. Based on elliptic curve digital signature algorithm (ECDSA), the scheme constructed linearly homomorphic authenticator, so that the data owner can verify the data integrity, and does not need to retrieve the entire data and thus dramatically reduces the communication and computation overhead. (2) An efficient public auditing scheme for data storage was proposed. It took advantage of Schnorr signature algorithm and homomorphic message authentication codes (MACs) to reduce the space used to store the verification metadata. The scheme also employed the random masking technique to make sure that the TPA cannot recover the users' outsourced data blocks.

Security proof shows that these schemes are secure. Furthermore, these auditing schemes can be extended to efficiently perform auditing for multiple different data file simultaneously, and execute dynamic operations efficiently. The experimental results have demonstrated our auditing schemes are much more light-weight than previous

auditing schemes. Due to pairing-free in the proposed auditing process, both the schemes can be effectively applied to distributed sensor network scenario, such as computing, storage, bandwidth, limited application environment.

2. Research on cloud data auditing mechanism supporting user revocability. (1) A privacy-preserving cloud data integrity verification scheme supporting user revocability was proposed. The scheme exploited the technique of bilinear aggregate signature to help current user audit the data which was sent to the cloud by all the previous users, and can satisfy the user's data transfer demand of large companies and organizations. Meanwhile cloud users can delegate a third party (TPA) to perform security auditing tasks as it is not economically feasible for them to handle the tasks by themselves. The scheme is provable secure in the random oracle model even when the cloud service provider conspires with revoked users. (2) A privacy-preserving integrity verification scheme for cloud storage supporting user revocability with the unidirectional proxy re-signature technique was proposed. In the proposed scheme, the proxy re-signature key was generated by the current data manager's private key and the former public key, which could not leak any information, thus it could realize transferring of ownership data caused by the users' revocability securely. Moreover, the random masking technique was employed to prevent the curious TPA from revealing the primitive data blocks. The scheme is also provable secure in the random oracle model even when the cloud service provider conspires with revoked users.

3. Research on efficient cloud data auditing supporting key-updating. An efficient privacy-preserving integrity verification scheme for cloud storage supporting key-updating was proposed. The scheme incorporates zero knowledge proof systems, proxy re-signatures and homomorphic linear authenticators. When the cloud user needs to update his key, instead of downloading the entire file and re-generating all the authenticators, the user can just download and update the authenticators. This approach dramatically reduces the communication and computation cost while maintaining the desirable security. The dissertation formalize the security model of zero knowledge data privacy for auditing schemes in the key-updating context and prove the soundness and zero-knowledge privacy of the proposed construction. We finally develop a prototype to evaluate the performance of our construction. Implementation results match the theoretical analysis and shows that the new scheme is good practicability. It is suitable for computing, storage and bandwidth limited applications in embedded cloud storage

systems.

4. Research on cloud storage data availability scheme supporting that type information of ciphertext can be modified. In this dissertation, except for the study of the integrity of the cloud storage data, we also further research the usability of the cloud storage data. To address the issue that different types of ciphertext stored in cloud server can be modified, the dissertation first point out that there exist two security flaws in Liu et al.'s scheme. Moreover, a dynamic type and identity-based proxy re-encryption scheme for cloud computing was proposed. The scheme not only keeps the traditional core function of PRE scheme, but also makes sure that the owner of ciphertext can modify the type information at any time. Compared with previous schemes, our scheme is more secure and practical.

Keywords: Cloud storage, integrity verification, user revocability, key update, proxy re-encryption, proxy re-signature

目 录

第一章 绪 论	1
1.1 研究工作的背景与意义.....	1
1.2 国内外研究现状和发展态势.....	3
1.2.1 云存储数据完整性验证.....	3
1.2.2 支持用户可撤销或密钥更新的云数据完整性验证.....	4
1.2.3 云计算环境下密文类型修改.....	5
1.3 论文的主要工作与组织结构.....	5
1.3.1 研究内容.....	6
1.3.2 论文的组织结构.....	7
第二章 预备知识.....	8
2.1 相关数学知识.....	8
2.1.1 双线性对映射.....	8
2.1.2 相关困难问题.....	8
2.2 基本工具及特征.....	10
2.2.1 同态标签.....	10
2.2.2 同态消息验证码.....	11
2.2.3 Shamir 秘密共享.....	11
2.2.4 代理重加密.....	12
2.2.5 代理重签名.....	13
2.2.6 群签名.....	13
2.3 复杂性理论基础.....	14
2.3.1 算法复杂性.....	14
2.3.2 问题复杂性.....	14
2.4 可证明安全.....	14
2.4.1 哈希函数.....	15
2.4.2 随机预言机模型.....	16
2.5 本章小结.....	16
第三章 高效的云存储数据完整性验证方案.....	17
3.1 DSN 云存储数据完整性验证	17
3.1.1 DSN 云存储数据完整性验证概述	17

3.1.2 相关研究进展.....	18
3.2 DSN 云存储数据完整性验证的应用模型	19
3.2.1 DSN 云存储数据的完整性验证框架	19
3.2.2 设计目标.....	21
3.3 DSN 云存储数据完整性自我验证方案	21
3.3.1 方案使用的主要技术以及优势.....	21
3.3.2 系统框架与风险模型.....	21
3.3.3 改进的 ECDSA	22
3.3.4 具体构造.....	23
3.3.5 安全性分析.....	24
3.3.6 支持数据动态操作.....	26
3.3.7 效率比较.....	26
3.4 DSN 隐私保护的云存储数据公开完整性验证方案	28
3.4.1 方案具体构造.....	28
3.4.2 方案安全性分析.....	31
3.4.3 支持批处理的扩展方案.....	33
3.4.4 方案的效率比较.....	35
3.5 本章小结.....	38
第四章 支持用户可撤销的云存储数据完整性验证方案.....	39
4.1 支持用户可撤销的云存储数据完整性验证研究概述.....	39
4.1.1 相关工作与研究动机.....	39
4.1.2 方案的系统模型.....	40
4.2 EDRPA 方案.....	44
4.2.1 方案的形式化定义与安全模型.....	45
4.2.2 方案的设计目标.....	48
4.2.3 EDRPA 方案.....	48
4.2.4 安全性证明.....	52
4.2.5 效率分析.....	54
4.2.6 实验分析.....	55
4.3 改进方案 PRRPA	56
4.3.1 基本模型.....	57
4.3.2 代理重签名.....	58
4.3.3 PRRPA 方案	58

4.3.4 PRRPA 方案的正确性与安全性分析	61
4.3.5 PRRPA 方案的性能分析	64
4.4 本章小结	65
第五章 支持密钥更新的高效云存储数据完整性验证方案	66
5.1 嵌入式云存储基本概述及相关研究工作	66
5.2 系统模型	69
5.2.1 嵌入式云存储数据完整性验证系统模型	69
5.2.2 方案定义及其安全性	70
5.2.3 设计目标	73
5.3 提出的方案	73
5.4 安全性分析	75
5.4.1 正确性	75
5.4.2 可靠性	76
5.5 性能分析与实验	77
5.5.1 参数选择	77
5.5.2 复杂性分析	77
5.5.3 实验结果	78
5.6 本章小结	82
第六章 密文类型可修改的云存储数据可用性方案	84
6.1 相关研究与系统模型	84
6.1.1 相关研究	84
6.1.2 系统模型	85
6.2 类型可修改的基于身份代理重加密方案	86
6.2.1 对 LUAN IBRAIMI 等人方案的回顾	86
6.2.2 类型可修改的基于身份代理重加密方案	87
6.3 对类型可修改的基于身份代理重加密方案的改进	87
6.3.1 LIU 等人提出方案的安全性分析	87
6.3.2 方案改进	90
6.3.3 正确性验证	91
6.3.4 安全性分析	91
6.4 本章小结	92
第七章 全文总结与展望	93
7.1 全文研究工作总结	93

目录

7.2 未来工作展望.....	94
致 谢	96
参考文献	97
攻读博士学位期间取得的成果	106
攻读博士学位期间参与的科研项目	108

第一章 绪论

1.1 研究工作的背景与意义

云计算是一种新兴服务模式，由于它能提供一个灵活、弹性且经济高效的基础设施^[1]，使得用户能够享受到诸多与地域无关的便捷数据服务，为用户带来了极大的便利。云存储服务是云计算中最重要的服务之一。事实上，面对日益增长的海量数据为了缓解本地数据存储与维护的压力，并能随时地进行访问或共享，越来越多的个人、公司与组织选择向例如 Dropbox、Google Drive、skyDrive、iCloud 等云存储服务商寻求帮助^[2]。

然而，用户选择将自己的数据外包存储在云服务器上后，数据便完全受控于云服务器，继而使得存储于云上的数据面临着严重的安全挑战^[3-6]。首先，云用户会担心他们存储在云服务器上的数据可能被未认证用户访问并滥用，因为在实际的云环境中，存在着众多恶意的敌手，他们出于各自的利益，试图窃取存储于云服务器上的用户数据。其次，云用户担心他们的云数据完整性遭到破坏^[7-8]，因为不管云服务提供商采用可靠程度多么高的措施，云数据丢失或损坏都不可避免。再次，为了经济利益，不诚实的云服务提供商通过删除访问频率低的数据或减少数据备份来节省存储开销。进一步地，云服务提供商出于维护声誉的考虑，可能在数据丢失后选择向用户隐藏事实，并宣称数据仍然正确地存储在云上。因此，如何确保云存储数据的完整性是一个很重要的研究课题。

云存储服务提供商应当长期确保云用户数据在云服务器上的完整性与可用性^[9]，并为云用户提供方便、可靠、满足实际可用性需求的完整性验证方法。传统的完整性验证方法需要首先从云端下载所有的数据，然后根据计算签名（例如 RSA^[10]）或是哈希值（例如 MD5^[11]）的结果来进行完整性判定。这种方式经常用于点到点存储系统^[12]，网络文件系统^[13-14]，Web 服务对象存储^[15]和数据库系统^[16]。但是使用这种模式同时也意味着用户只能在访问数据的同时进行数据的完整性验证，并不能很好地应用于类似于科研数据、地理信息、金融数据等大规模数据的完整性验证。因此，在不取回全部数据的前提下验证用户云存储数据的完整性是云数据完整性验证的基础。另外，对于云用户来说构建一个进行数据完整性检验的云环境可能是不可行或代价高昂的^[17-18]。为节省进行周期性存储完整性验证的通信资源和在线负担，云用户可以委托第三方执行安全审计任务。同时，云用户可能希望对第三方审计者(Third Party Auditor, TPA)保持数据的私密性。另外使用 TPA 进行云存储数据完整性验证，能够比较容易地将审计方案扩展到同时处理多个审计任务的情况（即批量审计）。批量审计不仅能使 TPA 同时执行来自不同验证

者的多审计任务，并能显著减少 TPA 的计算代价。这是因为将多个数字签名聚合后一起进行验证，能够减少大量的计算时间和存储空间。

随着云存储应用的深入发展，人们发现在类似于无线传感器网络、无线体域网、嵌入式设备等存储、计算、带宽资源受到限制的应用场景中，现有的云存储数据完整性验证方案并不适用。由于这些方案大多涉及到计算开销昂贵的双线性对操作，计算效率并不高，实际效果也不理想。因此，为提高云数据完整性验证的效率，使用基于离散对数问题（DL）的模指数验证等式来代替基于计算性 Diffie-Hellman（CDH）问题的双线性对验证等式，构造无双线性对的云存储数据完整性验证方案是一种有效的解决方法。此外，利用同态消息认证码可以减少存储认证信息的存储空间与审计证明响应信息的计算开销与通信代价。更进一步地，利用椭圆曲线数据签名算法（ECDSA）的特点^[19]构造线性同态验证标签，能够显著地减少通信代价和计算开销，从而实现在不取回整个数据的前提下，高效的进行云数据完整性验证。同时，通过哈希索引列表，方案可实现有效的数据动态操作。

在实际的云环境中，用户撤销在某些场景下很频繁。例如，一个组织或公司的数据管理者可能因为各种原因离职，或是密钥因为泄露或过期而需要更新。新用户希望对存储在云上的数据仍然能够进行完整性验证，但是却不希望将规模巨大的原数据取回并重新计算验证标签。人们很快注意到之前几乎所有的完整性验证方案都是固定用户在计算云存储数据的完整性验证标签。即这些完整性验证方案，要求在整个数据管理周期使用云存储服务的都必须是同一个用户。这是因为，云存储服务中数据的完整性验证标签是用户用自己的私钥签名生成，然后在完整性验证过程利用公开信息进行验证。这样的云存储数据完整性验证模式是不能满足用户撤销与密钥更新需求的。

随后一些支持用户可撤销的共享云存储数据完整性验证方案被提出，但是这些方案都涉及到群签名^[20]和广播加密技术^[21-22]，在效率上都不满足实际需求。进一步，使用代理重签名让云服务器代替新用户重新计算存储在云上的数据完整性验证标签，以方便让新用户对云存储数据进行完整性验证的方案被提出。但由于云存储数据验证标签的更改密钥由当前用户与已撤销用户通信生成，其代理重签名是双向的，存在云服务器与已撤销用户合谋泄露用户私钥的问题。因此，研究设计一个可抵抗云服务器与已撤销用户合谋的用户可撤销云存储数据完整性验证方案是一个非常重要的工作。在所设计的新方案中云存储数据验证标签的更改密钥应当由当前用户对已撤销用户的公钥处理生成，这样所构造的代理重签名方案是单向的将不存在密钥泄露问题。

此外，在实际的云存储应用中，密文类型信息的动态可修改性具有广泛的应用场景。比如：数据拥有者为节省本地存储开销，或方便数据在不同终端的灵活使用，将标记为“不可共享”或“可共享”类型的加密数据存储于云服务器。然后根据访问终端的实际需求对其进行相应修改，以方便数据的共享访问。

由上述内容可知，云数据的完整性验证保证云端数据的安全性，而云数据完整性验证的效率、用户可撤销、密钥更新、密文类型可转换等可用性问题与云数据的安全性问题一起制约着云存储应用的推广发展。因此，云数据的完整性与可用性研究具有重大意义。

1.2 国内外研究现状和发展态势

1.2.1 云存储数据完整性验证

Juels 等人在文献[23]中对数据完整性验证进行了先行研究，他们提出了一种名为数据可取回证明(Proofs of Retrievability, POR)协议。利用该技术可以对云存储数据进行有效的完整性验证，还可以提供一定程度的数据恢复功能。与此同时，Ateniese 等人在文献[24]中提出了数据可证明持有性(Provable Data Possession, PDP)协议，一个 PDP 协议只要添加一个前向纠错码就可以变成一个 POR 协议。在 PDP 方案中利用基于随机化数字签名算法(Randomized Digital Signature, RSA)的同态验证标签(homomorphic authenticator)和随机采样(sampling strategies)相结合的方式，验证者可以在不下载所有数据的前提下，公开地验证云端数据的完整性。尽管 PDP 首先考虑到了公开完整性检验，即利用一个第三方来代替用户自己对数据完整性进行检验，但是 PDP 中并未给出正式的安全性定义与安全性证明。之后，Shacham-Waters 在文献[25]中又提出了一种 POR 技术，该技术利用短签名(Boneh-Lynn-Shacham, BLS)进行云存储数据完整性验证，并首次给出了公开审计安全定义与安全证明。后续的工作都是沿袭 Shacham-Waters 的工作，有着相同的安全模型和威胁模型。在文献[25]的基础上，许多关于云存储数据完整性验证方案陆续被提出^[17,26-33]。

随着云计算的发展，支持隐私保护的第三方公开完整性验证方案、动态数据的可回取证明方案、支持批量审计任务的完整验证方案也得到了越来越多的关注。为了减小用户的计算负担，用户委托可信的第三方审计者(TPA)对存储在云服务器上的数据进行周期性地完整性验证。同时，云用户可能希望对 TPA 保持数据的私密性。2009 年，Wang 等人提出了支持 TPA 审计的确保数据安全的完整性验证方案^[27]和可公开验证方案^[17]。文献[27]考虑分布式情况下数据的动态存储，并将挑战

-应答协议应用于验证数据的正确性与错误定位。文献[17]以 BLS 签名技术为基础，同时引入了 RSA 结构，但该方案只支持部分动态操作，并且不提供隐私保护。2010 年，Wang 等人提出了一个改进的具有隐私保护功能的云存储数据公开完整性验证方案^[28]。数据拥有者在将数据和验证标签存储到云服务器后，在本地将其删除。在有审计需求时向 TPA 发出请求，由 TPA 执行审计任务，在 TPA 审计过程中，利用随机掩饰码技术保证 TPA 和云服务器不能够获取任何关于原数据的有用信息。2011 年，Wang 等人提出了一个利用 Merkle Hash 树^[34]（Merkle Hash Tree, MHT）实现了数据修改、插入、删除等动态操作的云存储数据公开完整性验证^[29]方案。但是 Xu 等人在文献[35]中指出该协议不能抵御恶意云服务器以及外部攻击者的攻击。云服务器即使通过了 TPA 的审计，也不能保证云用户存储数据的真实性和完整性。2013 年，Wang 等人又一次提出了改进方案^[26]，该方案对于来自外部的伪造攻击有着更强的抵抗力。然而，在一些实际应用中审计者无法从整个文件块中获取信息还是不够的。例如，审计者可以发起一个离线猜测攻击，从若干存储在云服务器的文件块中获取文件。在文献[36]中，Zhu 等人提出了一种协作的可证明数据持有方案，可支持多云服务器的批量审计，也可扩展支持动态审计^[37]。但是文献[36-37]都不支持多用户批量审计，因为在这两个方案中，用户生成数据验证标签使用了不同的参数，TPA 不能使用来自不同用户标签的线性组合进行审计。Wang 和 Zhu 提出了一系列的云存储数据完整性验证经典方案，但他们的方案都存在着繁重的计算开销，使审计性能大受影响^[9]，之后更多的支持数据动态操作的完整性验证方案^[38-40]被逐一提出。

1.2.2 支持用户可撤销或密钥更新的云数据完整性验证

自 Wang 等人与 Zhu 等人提出了一系列经典的具有保护隐私功能的云存储数据公开完整性验证方案^[26-28,36-37,39,41-44]后，人们很快注意到之前几乎所有的完整性验证方案都是固定用户在计算云存储数据的完整性验证标签。从实际应用考虑，一个实用的云存储数据完整性验证方案应该支持用户的动态撤销。

Wang 在文献[46]中首先引入了支持数据共享的云存储数据完整性验证问题，提出了一个基于群签名的用户可撤销的自我审计方案，接着提出了基于动态广播重签名方案和双向代理重签名的数据共享云用户可撤销公开完整性验证方案^[47-48]。随后，Yuan 等在文献[30]使用了一个类似的群签名技术提出了一个公开完整性验证方案。由于都涉及到群签名和广播加密技术，以上支持用户动态可撤销完整性验证方案的效率都难以满足实际需求。

2015 年 Wang 等在文献[49]中提出了一个高效的用户可撤销的公开完整性验证

方案，称为 Panda 方案。该方案借助代理重签名技术，将不同用户的数据块签名转换为当前用户签名形式，从而很好地满足了用户动态可撤销系统的云存储数据审计验证需求，这是此类问题当前的最优解决方案。但是 Wang 等在文献[49]中提到方案的局限性，提出云服务器与已撤销用户合谋可能会造成用户私钥的泄露，同时明确提出在下一步工作中希望通过一个多层次的单向代理重签名方案^[50]来解决这个问题。

1.2.3 云计算环境下密文类型修改

云存储中，代理重加密（Proxy Re-Encryption, PRE）^[51]技术可以保障用户数据存储在云服务器上的安全性与可共享性。该技术的核心思想是：数据拥有者以密文形式将数据存储在云服务器；数据拥有者可以委托云服务器对其存储的密文进行重加密并共享给其他用户。

最近 Liu 等人针对现实数据共享的应用在 Ibraimi 等人^[52-53]提出的基于类型和身份的代理重加密方案基础上提出了一种类型可修改的基于身份代理重加密方案^[54-57]。其方案不但实现了 PRE 中细粒度的密文共享，还解决了原方案密文类型信息静态，不能动态修改问题。Liu 等人敏锐地发现了密文类型信息动态修改的现实意义与应用场景，比如：数据拥有者为节省本地存储开销，或方便数据在不同终端的灵活使用，将标记为“不可共享”或“可共享”类型的加密数据存储于云服务器；然后根据访问终端的实际需求对其进行相应修改，以方便数据的共享访问。类型信息的动态修改就可以很好地适应此类应用。

Liu 等人的方案^[58]虽然具有很高的现实应用价值，并且其安全性分析声称具有与文献^[52]相同的安全性。但是在对其所提方案进行仔细研究后我们发现在完成密文类型信息的修改后并没有对密文类型信息进行验证。其方案存在两个安全漏洞：1、类型修改缺乏验证，攻击者可以随意修改类型标记，而不会被接收方发现。2、由于类型修改引起了新的条件性选择明文攻击问题。

1.3 论文的主要工作与组织结构

本论文在已有的云存储数据完整性方案的基础上，进一步研究设计更加高效的云存储数据完整性验证方案，并拓展了云存储数据环境中，基于云用户撤销、密钥更新、密文类型转换等可用性研究。在本节中，将对论文的主要研究内容与组织结构进行阐述。

1.3.1 研究内容

本论文针对云存储数据的完整性与可用性进行研究，将结合不同的密码技术以及各种应用环境对完整性验证方案进行分析和构造。具体工作如下：

1. 研究高效的云存储数据完整性验证方案。设计了一个高效的轻量级的云存储数据自我完整性验证方案与一个高效的云存储数据公开完整性验证方案。这两个完整性验证方案分别是基于椭圆曲线数据签名算法（ECDSA）与变型的 Schnorr 签名算法结合同态消息认证码技术构造而成的。由于完整性验证过程不含昂贵的双线性对操作，因此与现有的相关工作相比，该方案效率更高，能够有效应用于分布式传感器网络等计算、存储、带宽受限的应用环境。与此同时，在考虑云存储数据完整性验证效率的问题之外，还兼顾了云存储数据完整性验证的批量审计任务扩展、支持动态操作、支持数据隐私保护、严格的安全性证明等问题。

2. 研究支持用户可撤销的云数据完整性验证方案。基于双线性对的聚合签名算法提出了一个有效和安全的支持用户动态可撤销的公开完整性验证方案。并进一步基于单向代理重签名技术提出了一个具有隐私保护功能的支持用户可撤销的云存储数据公开完整性验证方案。方案能够安全地实现数据所有权的转移，有效的支持用户动态可撤销系统中对云数据完整性的验证需求。其中，基于单向代理重签名的用户可撤销完整性验证方案是基于 2015 年，Wang 等人文献[49]中的下一步工作设想完成的。其代理重签名密钥由当前用户私钥结合已撤销用户公钥生成，不存在私钥泄露问题，可以抵制由于已撤销用户和云服务商合谋而产生的攻击。

3. 研究高效的支持密钥更新的完整性验证协议。验证协议形式化了“零知识数据隐私”的安全模型，并提出了一种技术有效地解决了密钥更新问题。详细地说，利用简单的离散对数零知识证明使得 Shacham-Waters 协议[25]在密钥更新环境中不会泄漏任何外包数据的内容信息给验证者。同时，当云用户的密钥到期需要更新时，不需要再从云端下载他的数据。而是，从云端下载一个与整个文件相比非常小的认证作为替代。这样用户可以有效地更改验证标签，从而大大地降低了通信和计算代价。方案同时证明了在新的合理的安全模型下，新协议在随机预言机模型下是可证明安全。最后，由一个原型来评估框架的性能，实现结果与理论分析相吻合，表明该协议具有很好的实用性，适用于嵌入式云存储系统等计算、存储、带宽受限的应用环境。

4. 研究密文类型可修改的云存储数据可用性方案。设计了一个类型可修改的基于身份代理重加密方案。该方案在具有传统代理重加密方案的核心功能的同时，密文的拥有者还可以随时修改密文的类型信息。与相关工作相比较，方案解决了 2 个安全漏洞：1、类型修改缺乏验证，攻击者可以随意修改类型标记；2、类型修改

引起的新的条件性选择明文攻击问题。

1.3.2 论文的组织结构

第一章绪论。介绍相关的研究背景，概述本文将涉及的云存储数据完整性验证、支持用户动态可撤销与密钥更新的云存储数据完整性验证、云计算环境下密文类型修改的背景和研究现状，阐述本文的研究内容和组织架构。

第二章预备知识。介绍本文在设计云存储数据进行完整性验证时需要使用到的相关数学知识、基本工具、理论基础等。

第三章研究高效云存储数据完整性验证方案。首先分析了在分布式传感器网络等资源受到限制的应用环境下对云存储数据完整性验证存在效率需求，并实例化了分布式传感器网络云存储数据完整性验证方案的模型。接着，以此为基础，提出了一个适用于无线传感器网络轻量级的云存储数据自我完整性验证方案与一个支持隐私保护的公开完整性验证方案；并分别扩展支持数据动态操作与同时处理批量完整性验证任务。然后，进行安全性证明与效率分析。

第四章研究支持用户可撤销的云存储数据完整性验证方案。首先，形式化了支持用户可撤销的云存储数据完整性验证模型与安全风险模型。然后，在此基础上提出了一个支持用户可撤销的动态数据完整性验证方案 EDRPA，并对其进行了安全性分析与性能分析。接着，在 EDRPA 方案的基础上进一步提出了一种新的改进方案 PRRPA，同样对其进行了安全性分析与性能分析。

第五章研究高效的密钥更新的云存储数据完整性验证方案。首先对嵌入式云存储系统的密钥更新与零知识隐私保护问题进行概述。其次，给出嵌入式云存储系统数据完整性验证方案的系统模型以及安全模型。再次，给出具体的支持密钥更新与零知识隐私保护的实用嵌入式云存储公开完整性验证方案。最后，进行安全性分析并给出其实例化性能分析。

第六章研究密文类型可修改的云存储数据可用性方案。提出了一个改进的类型可修改的基于身份代理重加密方案。方案改进了相关方案的两个安全性漏洞：1、类型修改缺乏验证，攻击者可以随意修改类型标记；2、类型修改引起了新的条件性选择明文攻击问题。并对改进方案进行了安全性分析。

第七章全文总结与展望。先通过对论文所述的研究工作进行总结，在总结的基础上再提出下一阶段的研究展望。

第二章 预备知识

在本章中，将简要介绍研究相关的一些数学知识、基本工具与理论基础。具体有：双线性对映射、同态消息验证码、困难性问题、Shamir秘密共享、代理重加密、代理重签名、哈希函数、群签名与复杂性理论基础、可证明安全思想等。以上内容将帮助对云数据完整性与可用性研究中所设计的完整性验证方案的理解，为更好地进行下一步研究奠定基础。

2.1 相关数学知识

2.1.1 双线性对映射

双线性对映射可以通过椭圆曲线上的 Tate 对和 Weil 对实现^[59]。

定义 2.1 双线性对映射。

给定阶同样为素数 p 的三个循环群 G_1, G_2 和 G_T , g_1 是群 G_1 的生成元, g_2 是群 G_2 的生成元。则有双线性映射 $e: G_1 \times G_2 \rightarrow G_T$, 满足如下性质:

- (1) 双线性: 给定元素 $u \in G_1, v \in G_2$, 对任意 $a, b \in \mathbb{Z}_p$ 有 $e(u^a, v^b) = e(u, v)^{ab}$.
- (2) 非退化性: $e(g_1, g_2) \neq 1$.
- (3) 可计算性: 即是对于任意可能的输入都可以找到一个有效的算法, 能够有效的计算 e 。

若满足以上三条件, 则称映射 e 为双线性对或双线性映射^[60-61]。而且若 $G_1 \neq G_2$, 则称 e 为非对称双线性对^[61]; 如果 $G_1 = G_2$, 则称 e 为对称双线性对^[60]。

2.1.2 相关困难问题

密码学中, 安全性假设通常是大部分验证方案的安全性基础, 称之为困难问题。困难问题是指出在有限的条件与资源下无法通过计算被有效解决的问题, 下面对一些常用的困难问题进行介绍。

定义 2.2 整数分解问题 (Integer Factorization Problem, IF)。对于一个正整数 N , 求解其素数因子分解式 $N = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ 。因子分解式中, 所有的 e_i 为等于或大于 1 的整数, 所有的 p_i 为各不相同的素数, 这里 $1 \leq i \leq k$ 。若 ε 表示一个可以忽略的优势,

$$Succ_A^{IF} = \Pr[p \leftarrow A(n)] \leq \varepsilon \quad (2-1)$$

表示多项式算法 A 在多项式时间 t 内解决 IF 问题是计算上不可行或困难的。

定义 2.3 离散对数问题 (Discrete Logarithm Problem, DL)。群 G 是一个有限循环群, 给定两个元素 $P, a \in G$, P 为群 G 的一个生成元, 计算 $x \in Z_p$ 使得等式 $a = xP$ 成立^[62]。若 ε 表示一个可以忽略的优势,

$$\text{Succ}_A^{DL} = \Pr[x \leftarrow A(P, a)] \leq \varepsilon \quad (2-2)$$

表示多项式算法 A 在多项式时间 t 内解决 DL 问题是计算上不可行或困难的。

通常采用Pollard的 ρ -算法与Baby-Step/Giant-Step算法^[63]来求解任意循环群中的 DL 问题, 其中Pohlig-Hellman算法^[64]主要适用于具有光滑阶的循环群。此外, 群 G 中元素规模的大小也常常影响到 DL 问题的困难性, 因此 DL 问题的困难性在不同的群中, 也是不尽相同的。

定义 2.4 计算Diffie-Hellman问题(Computational Diffie-Hellman Problem, CDH)。群 G 是一个有限循环群, 给定 $P, aP, bP \in G$ (其中整数 a, b 是 Z_p 中的两个未知数), P 为群 G 的一个生成元, 计算 $abP \in G$ 。若 ε 表示一个可以忽略的优势,

$$\text{Succ}_A^{CDH} = \Pr[abP \leftarrow A(P, aP, bP)] \leq \varepsilon \quad (2-3)$$

表示多项式算法 A 在多项式时间 t 内解决 CDH 问题在计算上不可行或困难的。

定义 2.5 判定Diffie-Hellman问题(Decision Diffie-Hellman Problem, DDH)。群 G 为一个有限循环群, 给定 $P, aP, bP, cP \in G$ (其中整数 a, b, c 是 Z_p 中的三个未知数), P 为群 G 的一个生成元, 判定 $ab = c \pmod{P}$ 是否成立。若 ε 表示一个可以忽略的优势,

$$\text{Succ}_A^{DDH} = \Pr[ab \stackrel{?}{=} c \pmod{P} \leftarrow A(P, aP, bP, cP)] \leq \varepsilon \quad (2-4)$$

表示多项式算法 A 在多项式时间 t 内解决 DDH 问题在判定上是不可行或困难的。

在上述问题中, DL问题、CDH问题、DDH问题三者之间的困难关系在难易程度上可以这样进行解读: $DL \rightarrow CDH \rightarrow DDH$, 即如果DL问题是可解决的, 那么必然有着可被解决的CDH问题和DDH问题; 如果CDH问题是可被解决的, 那么DDH问题也可以被解决, 但是反之却不一定成立。需要特别指出的是, CDH问题在特定的群中等价于DL问题^[64]。例如, A.Joux与K.Nguyen构造出的一类椭圆曲线^[65], DL问题和CDH问题在一些子群上是困难的, 同时也进行了等价证明, 而DDH问题在这些子群上却是可解决的, 这样的群被称为理想群(Gap Diffie-Hellman Group, GDH)。

定义 2.6 双线性Diffie-Hellman问题^[66] (Bilinear Diffie-Hellman Problem, BDH)。群 G 为一个有限循环群, 给定 $P, aP, bP, cP \in G$ (其中整数 a, b, c 是 Z_p 中的

三个未知数), P 为群 G 的生成元, 计算 $e(P, P)^{abc}$ 。若 ε 表示一个可以忽略的优势,

$$Succ_A^{BDH} = \Pr[e(P, P)^{abc} \leftarrow A(P, aP, bP, cP)] \leq \varepsilon \quad (2-5)$$

表示多项式算法 A 在多项式时间 t 内解决BDH问题在计算上是不可行或困难的。

需要注意的是, 在解决CDH的同时, 就能够解决BDH问题。如果此时群 G 上的CDH问题是可解的, 那么给定 $P, aP, bP \in G$, 可以求出 $abP \in G$, 同样可以计算出 $e(P, P)^{abc} = e(abP, cP)$ 。

定义 2.7 s -计算性Diffie-Hellman问题(s -CDH Problem)。群 G 为一个有限循环群, 给定 $Q, Q^x, Q^{x^2}, \dots, Q^{x^s} \in G$, 其中 Q 为群 G 的一个元素, 而 x, s 为属于 Z_p 的两个未知整数, 计算 $Q^{x^{s+1}} \in G$ 。若 ε 表示一个可以忽略的优势,

$$Succ_A^{s-CDH} = \Pr[Q^{x^{s+1}} \leftarrow A(Q, Q^x, Q^{x^2}, \dots, Q^{x^s})] \leq \varepsilon \quad (2-6)$$

表示多项式算法 A 在多项式时间 t 内解决 s -CDH问题在计算上是不可行或困难的。

定义 2.8 s -计算性Diffie-Hellman逆问题(s -CDHI Problem)。群 G 为一个有限循环群, 给定 $Q, Q^x, Q^{x^2}, \dots, Q^{x^s} \in G$, 其中 Q 为群 G 的一个生成元, 而 x, s 为属于 Z_p 的两个未知整数, 计算 $Q^{1/x} \in G$ 。若 ε 表示一个可以忽略的优势,

$$Succ_A^{s-CDHI} = \Pr[Q^{1/x} \leftarrow A(Q, Q^x, Q^{x^2}, \dots, Q^{x^s})] \leq \varepsilon \quad (2-7)$$

表示多项式算法 A 在多项式时间 t 内解决 s -CDHI问题在计算上是不可行或困难的。

此外, 基于双线性的其他困难问题^[67-69], 还有比如弱Diffie-Hellman问题, 强Diffie-Hellman问题, 选择目标CDH问题等。

2.2 基本工具及特征

此节将简述: 同态标签、同态消息验证码、Shamir秘密共享、代理重加密、代理重签名、环签名、群签名、盲签名等本文所涉及到的基本工具以及其特征。

2.2.1 同态标签

同态标签(homomorphic authenticators or homomorphic verifiable tags)^[24], 允许公开验证者在不用下载完整数据的前提下, 验证数据的完整性。同态标签作为基本工具现今已广泛应用于云存储数据的完整性验证, 例如文献[2,8,17,24-25,27,39,42,46,70-74]。一个合法的同态标签应满足以下特性:

设签名者的公私密钥对为 (pk, sk) , 数据块 (block) $m_1 \in Z_p$ 、 $m_2 \in Z_p$ 的标签 (tags或signature) 分别表示为 σ_1 和 σ_2 。

1. 不可伪造性 (Unforgeability): 即只有持有密钥的用户可生成合法标签。
2. 数据块简化验证 (Blockless Verifiability): 即已知 σ_1 与 σ_2 , 随机数 $\alpha_1, \alpha_2 \in Z_p$ 和数据块 $m' = \alpha_1 m_1 + \alpha_2 m_2 \in Z_p$ 验证者可以在不知道 m_1 和 m_2 的情况下, 验证数据块 m' 的正确性。
3. 非扩展性 (Non-malleability): 即使已知 $m_1, m_2, \sigma_1, \sigma_2$, 随机数 $\alpha_1, \alpha_2 \in Z_p$ 和数据块 $m' = \alpha_1 m_1 + \alpha_2 m_2 \in Z_p$, 仍然无法通过线性结合 σ_1 与 σ_2 的方式来生成一个基于数据块 m' 的合法数据验证标签, 这是因为非法用户不可能拥有私钥 sk 。

数据块简化验证的实质指的是在云存储数据完整性验证中, 验证者不回检所有的云存储数据, 而是通过挑战与应答协议 (Challenge-Response), 使用所有数据块的线性组合与相对应的数据标签的线性组合来验证挑战所选中的云存储数据的正确性和完整性。验证的非扩展性指任何攻击者在无法获取私钥的情况下, 都不能够在已有的数据块线性组合验证标签的基础上构建新的有效数据标签以通过验证。

2.2.2 同态消息验证码

同态消息验证码 (Homomorphic MACs) 是Agrawal与Boneh^[75]提出的, 用于在传输过程中对数据的完整性进行验证。同态消息验证码允许在不持有私钥的状态下, 在中间节点通过输入数据块的消息验证码, 得到一个有效的由此数据生成新的消息验证码作为输出。假设已知数据块 $m_j = (m_{j,1}, \dots, m_{j,n}) \in Z_p^n$, $a = (a_1, \dots, a_n)$ 是一组伪随机生成器 (Pseudo-random generator) 利用私钥 k_{prg} 生成的伪随机序列, b_j 是一组由伪随机函数 (Pseudo-random function) 利用私钥生成的伪随机数。这个数据块的同态消息验证码可由下式计算生成:

$$t_j = \sum_{i=1}^n a_i m_{j,i} + b_j \in Z_p \quad (2-8)$$

即已知 t_1 与 t_2 , 中间节点可在密钥对 (k_{prf}, k_{prg}) 未知的情况下, 为新数据块 $m' = m_1 + m_2$ 计算一个合法的消息验证码: $t' = t_1 + t_2$ 。

2.2.3 Shamir 秘密共享

Shamir秘密共享 (Shamir secret sharing)^[76]是由Shamir首先提出来的。在 $Shamir(s, t)$, ($s \geq 2t - 1$) 共享方案中一个秘密 τ 被分成 s 块, 该秘密 τ 可以由其中的任意 t 块数据恢复出来。但其中任意的 $t - 1$ 块数据都不会泄露秘密 τ 的任何信息。

利用平面上的 t 个点可决定一个唯一的 $t - 1$ 次多项式 (polynomial) 是Shamir秘

密共享的基本思想。设 $t - 1$ 次多项式的形式如下：

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1x + a_0 \quad (2-9)$$

其中， $a_{t-1}, \dots, a_1 \stackrel{R}{\leftarrow} Z_p^*$, $\tau = a_0$, 秘密的每一块可为多项式上的点来表示，例如：对于 $1 \leq i \leq s$, 点可表示为 $(x_i, f(x_i))$ 。秘密 τ 可由拉格朗日多项式公式 (Lagrange polynomial interpolation) 通过任意的 t 个点计算多项式 $f(x)$ 求解，从而恢复出秘密 τ 的本来面目。Shamir秘密共享目前广泛应用于密钥管理^[77]与安全多方计算 (secure multi-party computation) ^[78]中。

2.2.4 代理重加密

在此节中，将简要的介绍一下代理重加密的概念，并对代理重加密的一些性质做简单描述。

代理重加密是由Blaze等人^[51]首先在 1998 年欧洲密码学会议上提出的。在代理重加密方案中，半可信代理 (proxy) 可以利用其所控制的额外信息，将消息 m 的Alice (代理者) 的公钥加密状态转换成同一消息 m 的Bob (被代理者) 公钥加密的状态。在密文转换过程中proxy既不能获取任何关于消息 m 的信息，也不能借此得到任何使用Alice或Bob的公钥加密消息的任何信息。代理重加密主要被划分形式成两类形式^[51]。根据密文可转换方向被划分为：(1) 双向的代理重加密，Alice 与 Bob 之间的密文，proxy 使用同样的信息就可以对其进行相互转换。(2) 单向的代理重加密，proxy 使用一种信息只能将 Alice 的密文转换成 Bob 的密文，但不能把 Bob 的密文转换成 Alice 的密文。根据密文被允许转换的次数划分的另一种形式：(1) 单跳的，密文仅仅只能被转换一次，例如从 Alice 的形式转换为 Bob 的形式；(2) 多跳的，密文可被转换任意次数。从理论上来说多跳的代理重加密方案在密文转换的次数方面没有限制要求。

文献[79]总结了代理重加密方案的九条重要属性，定义如下：

1、单向性 (unidirectionality): 对一个单向的代理重加密方案来说，代理所拥有一部分额外信息，能将密文从一方转换到另外一方，但反向转换是不成立的。在双向代理重加密方案中，代理所拥有的额外信息能够完成代理者密文与被代理者密文之间的互相转换。显然，单向代理重加密方案要优于双向的代理重加密方案，因为后者可以由两个转换方向相反的前者组合而成，但前者却不能构造后者。一个单向的代理重加密方案在设计上相较于一个双向的代理重加密方案要困难。

2、多跳性 (Multi-use): 一个密文可被转换多次的多代理重加密方案被描述为多跳重加密方案，而在单跳代理重加密方案中，密文只能被转换一次。

3、秘密代理 (Private proxy): 对一个秘密代理代理重加密方案来说，代理

用来转换密文的额外信息是保密的。而在公开代理的代理重加密方案中，代理的相关额外信息在代理进行密文转换的过程中是处于公开状态的。

4、透明性（Transparent）：授权者同样能够生成由代理转换的密文，这样的代理重加密方案叫透明的代理重加密方案。

5、密钥最优（Key-optimal）：对于一个密钥最优代理重加密方案来说，用户需要保护和存储的秘密只是一个非常小的常数。

6、非交互性（Non-interactivity）：在非交互的代理重加密方案中，生成代理所需额外信息的过程不需要被代理者参与。

7、非传递性（Non-transitivity）：对于一个具有非传递性属性的代理重加密方案来说，密文转换的功能不是由代理自身所完成的。

8、暂时性（Temporary）：对于一个具有暂时性的代理重加密方案来说，密文转换功能可以由随时地指定的代理来完成。

9、抗合谋攻击（Collusion-resistant）：在可抗合谋代理重加密方案中，代理不能和被代理者合谋从而得到代理者的私钥信息。

2.2.5 代理重签名

代理重签名（Proxy re-signatures），是由Blaze等人^[51]首先提出的。代理重签名允许一个半可信（semi-trusted）的代理（proxy）对不同的用户（例如Alice和Bob）的签名进行转换。即是，代理可将Alice的一个数据块上的签名转化成Bob对同一个数据块上的签名。同时，在代理转换签名的过程中，并不能获取任何用户的私钥，代理亦不能代替任何的用户计算生成新的任意一个数据块上的标签。在本文中，我们使用代理重签名技术实现了用户撤销过程中数据所有权的安全移交。

2.2.6 群签名

群签名（Group Signatures）的概念是由Chaum和Heyst等人^[20]首先提出来的。群签名主要确保在群组用户中签名者的身份隐私。例如，验证者被说服相信的一个消息被某一个群组用户签名时，却无法分辨这个群组用户的真实身份。在性质上群签名与环签名相类似，但是存在一定差别，群签名方案中存在着一个群组管理员（group manager），他可以在某些特殊情况下追踪（trace）群签名消息所对应的用户真实身份。Boneh等人^[80]基于 $q - SDH$ 构建了一种短签名，在此方案中，短签名的长度为一个固定值与群组的用户数量无关。

2.3 复杂性理论基础

在了解攻破某个密码学方案实际难易程度的研究中，计算复杂性理论有着重要价值^[81]。在设计算法的过程，往往希望能够兼顾高效与安全两个方面。就安全方面来说，设计者希望较高的计算复杂性能够保证算法的安全性。

2.3.1 算法复杂性

算法复杂性指运行一个算法所需要耗费的计算资源，即计算复杂度^[82]。在实际中，统计精确的计算时间是相当困难的，因此通常会近似的用符号 $O(\cdot)$ 来替代关于输入规模 n 的函数。

定义 2.9 (多项式时间算法)：设两个正整数函数分别由 $f(n)$ 与 $g(n)$ 表示，假设存在一个常数 C ，对于所有的 $n > N$ （ N 为给定的自然数），始终满足 $|f(n)| \leq C|g(n)|$ ，则可记函数 $f(n)$ 计算复杂度为 $O(g(n))$ 。

举例：如果一个算法的多项式时间复杂性表示为 $2n^4 + 7n^2 + 5n + 3$ ，那么它的主要计算复杂性体现于 n^4 部分，可以使用 $O(n^4)$ 表示。类此，若 $p(n)$ 是次数为 n 的多项式时间复杂度多项表达式，则时间复杂度为 $O(p(n))$ 的算法叫做多项式时间算法，相对的指数时间算法指其余不能这样表示的时间复杂度算法。

2.3.2 问题复杂性

通常情况下，多项式时间算法被认为是有效的算法，而非多项式时间算法则被认为是无效的算法。如果一个问题被称为不可解困难问题，通常是指这个问题不能使用多项式时间算法求解。按求解代价对数学问题进行分类，可分为三类^[83]： P 、 NP 、 NPC (NP completeness)。在多项式时间内可以求解的问题称为 P 类问题；不确定多项式时间可以求解的问题被称为 NP 类问题， NP 类问题是可由多项式时间算法来验证语言类的，并保持 $P \subseteq NP$ 。 NPC 问题是 NP 中最困难的问题。若是任意一个 NPC 问题在多项式时间内可解，那么 NP 中的每个一困难问题都可以在多项式时间求解。一般来说，密码系统的安全性皆可归约成数学基础中的某个 NP 问题^[84]。

2.4 可证明安全

形式化的可证明安全是现代密码学理论中非常重要的一部分。形式化证明可以从本质上对一个密码学方案进行安全性分析，并给出一个定性的结果。可证明安全思想由Goldwasser等人^[85]于 1984 年首次提出。现今，越来越多的密码学研究者已经意识到可证明安全思想的重要性。这是由于（1）信息保护对密码方案安全

性的要求越来越高，安全证明对于密码方案变得必不可少；（2）攻击技术越来越强大，设备计算能力的提高必然带来攻击者攻击能力强化，以前很多被认为是安全的方案都被逐步破解。因此，面对日益增加的可证明安全分析的必要性需求，现今的密码学安全方案都应具备相应的安全性证明以确保其安全性。

可证明安全的基本证明思路类似于数学上的反证法，具体描述如下：事先假设敌手可以以一个不可忽略的概率攻破某安全方案，然后通过敌手与预言机之间的交互过程将敌手在一定时间与概率内攻破某安全方案的能力规约为敌手攻破某个预设困难性问题所需要的能力。这样敌手在攻破这个安全的方案的同时，也必然可以解决预设的困难性问题。这就与困难性问题的不可解决性相违背。

2.4.1 哈希函数

哈希函数在密码学中有着非常广泛的应用，它通常被称为杂凑函数或散列函数。由于哈希函数的有着固定的输出长度，因此使用哈希函数对不同比特串长度的消息进行处理，可以得到相同长度比特串的消息。在密码学方案中，使用哈希函数大幅度压缩消息的长度可以有效减少存储空间、提高计算效率。另外利用哈希函数的单向性与抗碰撞性可有效的对数据的完整性与认证性进行保护，提高方案的安全性。

定义 2.10 哈希函数是一种具有将任意长度比特串转变成固定长度散列值能力的确定性函数。设一个输出为 n 比特长度的哈希函数为 $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ ，函数 H 应具备下述五种安全特性^[86]：

1. 散列性：针对任意的输入 x ，输出 $H(x)$ 的值在计算上与均匀分布在区间 $[0, 2^n]$ 中的二进制串值是不可区分的。
2. 抗弱碰撞性：针对一个任意的输入 x 和相应的输出 $H(x)$ ，要想找到另外的一个不同输入 y 来满足 $x \neq y$ 的情况下，对应的输出 $H(y)$ 与 $H(x)$ 相等，这在计算上是不可行的。
3. 抗强碰撞性：对于任意的输入 x 和 y 满足 $x \neq y$ ，使得公式 $H(x) = H(y)$ 成立，是计算不可行的。
4. 单向性：若已知一个哈希值 h ，要找出一个输入值 x ，使得公式 $H(x) = h$ 成立的，是计算不可行的。
5. 有效性：对于任意的输入 x ，都可以在当前软硬件设施条件下在关于 x 长度规模的低阶多项式时间以内计算出 $H(x)$ 的值。

2.4.2 随机预言机模型

在对密码方案的安全性讨论中，通常会假设一个伪随机函数就是哈希函数。Fiat和Shamir首先利用这种假设哈希函数的方法证明了大整数分解问题与签名方案的安全性是等价的^[87]。1993年Bellare与Rogaway进一步研究并提出了随机预言机模型(Random Oracle Model, ROM)^[88]的概念。ROM模型给密码学方案安全证明构造了一个通用框架。ROM模型虽然不具有对现实情况进行完全模拟的能力，但其仍不失一个即实用又合理的方法。可证明安全理论在提出ROM模型之后开始不在停留于纯理论研究层面，进入一个高速发展的阶段。

随机预言机可以形容成一个黑匣子，输入一个 x ，就能得到一个输出 $H(x)$ 与之相呼应。在安全模拟过程，挑战者和攻击者都可以询问预言机；而在具体实施过程中密码学方案一般会采用一个实际的哈希函数来替换随机预言机。这是因为，现实中的散列函数是一个确定性函数，并不能保持输出均匀分布的同时保持输出无碰撞。随机预言机定义如下：

定义 2.11 输出长度为 n 的哈希函数 $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ 若满足以下特性被称为一个理想的随机预言机^[86]：

1. 确定性：对于相同的输入 x 、 y ，输出也相同 $H(x) = H(y)$ 恒成立。
2. 均匀性：理想随机预言机的输出与均匀分布在取值空间的值是不可区分的。
3. 有效性：给定输入 x ，能够在多项式时间计算出对应的 $H(x)$ 。

随机预言模型的使用在密码学界存在一定的争议。在最初的应用阶段，ROM模型下的可证安全方案被很多学者认为不存在安全漏洞。但是随后Canetti等人^[89]对此看法提出了置疑，他们指出如果随机预言机被使用真实的散列函数替换，该模型下可证安全的方案就可能会受到影响，从而不再安全。因此，一部分学者开始采用标准模型下的可证安全方案，许多标准模型下可证安全的密码学方案陆续被提出^[90]。对比标准模型下的可证安全方案与ROM模型下的可证安全方案，标准模型下可证安全的方案无疑更为安全，但其效率与可扩展性要差一些。Pointcheval等人^[91]明确指出，ROM模型在实际安全性度量方面，仍是极其有效的手段，如果ROM模型下方案中所结合的哈希函数没有漏洞，整个方案的安全性仍能够由该安全证明确保。

2.5 本章小结

本章介绍了论文相关的数学知识、基本工具、理论基础、安全证明思想等预备知识。这些知识为本文密码方案设计的基础，对本章的阅读有助于理解后续内容。

第三章 高效的云存储数据完整性验证方案

本章研究的主要内容是云存储数据完整性验证的高效审计问题。随着云计算与分布式传感器网络（Distributed Sensor Networks, DSN）（尤其是无线传感网络）的紧密结合^[92]，云存储服务在此类场景的推广应用正迅速被人们接受。将传感器数据存储在云上能够极大地方便对传感器数据的访问与管理，但与此同时又引入了新的挑战^[92]，即在外包数据完整性验证需求的基础上又提出了完整性验证效率要求。而许多云存储数据完整性验证经典方案大多需要计算开销较大的双线性运算，这些方案由于验证效率问题并不适用于计算能力、存储能力、带宽等资源受到极大限制的分布式传感器网络。从提高云数据完整性的验证效率出发，本章对无双线性对的云数据完整性验证方案进行了研究。首先，基于椭圆曲线数字签名算法（ECDSA）提出了一个无双线性对的轻量级的云存储数据自我完整性验证方案。接着利用变型的 Schnorr 签名算法与同态消息认证码 MACs^[75]技术构建了一个无双线性对的高效的公开完整性验证方案。效率分析与实验结果表明，无双线性对的云存储数据完整性验证方案相比需要双线性对计算的云数据完整性验证方案更加的轻量级和高效率，适用于分布式传感器网络环境。

本章内容由五部分构成：首先，3.1 节概述了分布式传感器网络云存储数据完整性验证研究；其次，3.2 节介绍了分布式传感器网络中的云存储数据完整性验证应用模型；再次，3.3 节提出了一个轻量级的自我完整性验证方案；然后，3.4 节提出了一个高效的公开完整性验证方案；最后，3.5 节对本章内容进行小结。

3.1 DSN 云存储数据完整性验证

本节将介绍 DSN 云存储数据完整性验证的基本情况及其研究现状。DSN 云存储数据完整性验证与传统云存储数据完整性验证相比，主要区别在于：相对于计算机终端用户，DSN 数据托管用户拥有的资源极其有限，不能负担大量的计算任务、存储过大的数据、占用过多的带宽，需要尽可能地减少 DSN 用户在数据完整性验证方案中的计算开销与通信代价。

3.1.1 DSN 云存储数据完整性验证概述

现今，DSN 已经大量应用于人们的社会生活^[93-94]。随着 DSN 的广泛应用，DSN 云存储数据的安全和隐私保护变得越来越重要^[95-96]。DSN 数据管理者常常需要搜集大量的数据并将其存储到云服务器上。正如我们所见，云计算是一个合适

的可替代计算模型，因为它能提供一个灵活、有弹性、且经济高效的基础设施^[1]。因此，将大量的 DSN 数据存储到云服务器将是一个最合适的选择^[92]。云存储作为云计算的一个重要服务，允许云用户通过数据外存将数据从本地存储系统转移到云上。因此，DSN 数据管理者能够从本地数据存储和维护，这些昂贵的负担中解放出来，并将精力集中于核心业务上，而不是处理大量存储软硬件以及相关的人员聘用与维护等日常事务。

尽管将 DSN 数据存储在云上会给 DSN 数据管理者带来极大的便利，但是 DSN 数据管理者也必将面对与普通云存储用户同样的安全性挑战^[4-6,101]。(1) 虽然云服务提供商能够提供安全性更高的存储设备，但是海量的数据存储在云服务器上使得数据更容易遭受攻击者的主动攻击。在这方面已有许多关于资料托管的安全事件研究^[97-99]。(2) 虽然云服务提供商能够提供更高级别的可靠性措施，但最近的报告表明云存储数据的丢失仍然不可避免。(3) 对云服务提供商来说，出于一些利益原因，他们可能不会真实的反应用户数据的存储情况，例如：云服务提供商可能因为利益原因删除一些用户很少访问的数据或减少数据备份的数量，云服务提供商可能会在用户数据损坏后通过隐瞒事实来维护他的声誉。因此，将 DSN 数据存储在云上完整性验证同样必不可少。

与普通的云存储用户一样，DSN 数据管理者希望在不取回全部数据的前提下对存储在云服务器上的数据进行完整性验证。并能够通过第三方审计者（TPA）执行安全审计任务，因为这样可以节省进行周期存储完整性验证的通信资源和在线负担。同时，DSN 数据管理者希望在 TPA 进行完整验证的过程中能够保护自身的数据隐私。此外，DSN 数据管理者考虑到自身的设备与环境因素，他希望在计算云存储数据完整性验证标签的开销上与通信代价上尽可能的小，所使用的完整性验证方案尽可能的轻量级。

3.1.2 相关研究进展

迄今为止，一系列不需要取回整体数据的云存储数据完整性验证方案^[17,23-26,39,73-74,102]已陆续被提出。然而，其中一些方案^[17,24-25]并没有考虑保护用户的数据隐私。用户的数据可能被一些好奇的敌手恢复出来。这些缺点将进一步影响这些方案在云计算环境下的安全性。从保护数据隐私的角度出发，用户可以委托一个第三方审计者 TPA 来保证他们存储数据的安全，但同时他们又不希望 TPA 的审计过程中由于未经授权的信息泄漏对他们的数据安全造成新的威胁^[100]。2013 年，Wang 等^[26]提出了一项基于双线性对的保护隐私的云存储数据公开完整性验证方案。方案依托同态认证技术和随机掩饰码技术实现了隐私保护和公开完整性验

证功能，并利用聚合签名技术完成了数据完整性验证的同时批量任务处理。Wang 等^[26]声称他们的方案是高效的，但是其方案涉及到计算开销巨大的双线对运算，并不能满足分布式传感器网络等资源受到极大限制的应用场景。因此，设计不使用双线性对运算的轻量级的具备隐私保护功能的云存储数据完整性验证方案将是当前研究的一个重要课题。

3.2 DSN 云存储数据完整性验证的应用模型

在提出实际方案之前，需要对 DSN 云存储数据完整性验证的应用模型作一个简单地分析。通过分析，具体化 DSN 云存储数据完整性验证需求，并归纳出适用于 DSN 应用场景的安全高效的数据完整性验证方案设计目标。

3.2.1 DSN 云存储数据的完整性验证框架

在本节中，将结合实际的分布式传感器网络提出一个适用于分布式传感器网络的云存储数据完整性验证框架。在一个分布式传感器网络应用场景中，出于简化的目的，假设 Sink 节点搜集完毕传感节点数据后，存在一个 DSN 数据管理者来处理和传输 DSN 数据到云服务器。因为 DSN 数据管理者没有附加的计算资源，所以他只能利用 Sink 节点有限的能力实现 DSN 数据的安全存储。在这个场景中，DSN 数据管理者是 DSN 的一部分，但对于云服务提供商，则可以被看作一个特殊的云用户。具体来讲，假设 Pob 是一个 DSN 的数据拥有者，他的任务主要是搜集传感器网络数据进行处理并以此完成多种面向客户的应用服务。但是他又没有足够的预算来购买或租用专业的存储设备，于是他倾向于将他的数据外存到云服务提供商(CSP)提供的云服务器上。与此同时，他会担心以下几个问题：

1. 当 DSN 数据存储到 CSP 提供的云服务器之后，他将无法采用传统的控制方法来控制他的数据，他担心 CSP 在数据损坏后会拒绝承认或者对数据进行恶意篡改。因此，他需要能够随时地验证存储在云服务器上的数据是完整有效的。
2. Pob 的主要工作是响应传感器网络所提出的各种应用，所以他需要一个有效的完整性验证方案可以让第三方审计者来替他完成周期性的数据完整性验证任务。
3. 在实际的应用中，CSP 与 TPA 会诚实的执行存储任务与审计任务，但是同时也是好奇的，他可能会想知道用户到底存储了什么数据在云服务器上。但是，Pob 想要保证他所存储的数据是机密的，除了他自己和他指定的授权者能得到真实的数据以外，其他的人都不能得到真正的存储内容。因此，他需要确保这个存储模式能对他所存储的数据进行隐私保护。

4. Pob 依靠云服务器来进行数据的存储和管理。为实现多种应用目的，他需要能够和云服务器进行动态的数据交互，访问并更新他存储的数据。

5. Pob 的 DSN 可能分别布置于多个地域。为简化管理，他需要 TPA 能够同时处理来自多个 Sink 节点（DSN 数据管理者）的审计任务。

6. 在 DSN 数据云存储模型中 DSN 数据管理者利用 Sink 节点有限的能力计算数据验证标签，并上传数据与相应的验证标签到云服务器。因此 DSN 数据管理者希望云数据的完整性验证方案应当尽可能的轻量级。

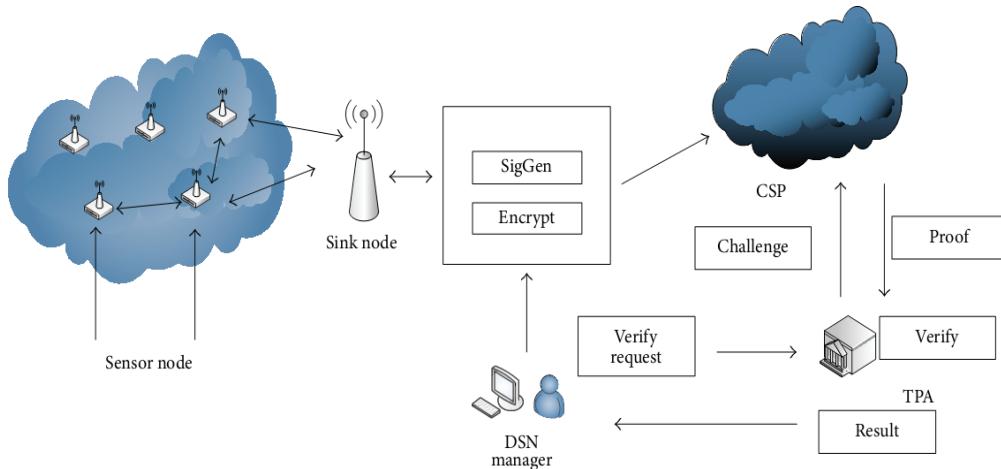


图 3-1 基于 DSN 的云存储数据完整性验证框架

如图 3-1 所示，DSN 数据云存储模型包括以下组成部分：传感节点（Sensor node）、汇聚节点（Sink node）、云服务提供商(CSP)、DSN 数管理者(DSN manager)、第三方审计者(TPA)。其中 Sensor node 负责从目标搜集信息，并将其传送给 Sink node，Pob 是一个分布式传感器网络的数据拥有者，他会分配一个 DSN 数据管理者来签名和加密数据。紧接着，DSN 数据管理者将生成的外包存储数据和相应的数据标签传输到 CSP 提供的云服务器上，同时删除本地存储的数据。如果这个 DSN 数据管理者想要验证存储在云服务器上数据的完整性，他需要向 TPA 提出一个询问。TPA 在收到这个询问之后将验证这个询问是否合法，如果是合法的，TPA 将生成一个挑战信息来向云服务提供商索要存储在服务器上的数据验证标签。云服务提供商在收到这个挑战询问之后，将生成的响应证明和数据验证标签返回给 TPA。TPA 验证这些信息，然后返回一个有效的认证结果给 DSN 数据管理者。最后，DSN 数据管理者将审计结果告诉 Pob。

3.2.2 设计目标

为确保本章适用于 DSN 应用场景的云存储数据完整性验证方案在前述框架下是安全高效的，方案应当达到下面的安全和效率目标。

1. 公开验证：在不取回整个数据或是增加用户在线负担的前提下，允许 TPA 验证云存储数据的完整性。
2. 存储正确性：确保不存在不诚信的云服务器在存储数据块损坏的情况下通过 TPA 的审计验证。
3. 隐私保护：确保 TPA 不能够在审计过程中以任何途径获取 DSN 数据管理者的数据内容信息。
4. 批量审计：TPA 能够在审计过程中，满足大量不同的 DSN 数据管理者同时提出的审计验证任务请求，并安全、高效的执行数据审计验证任务。
5. 验证轻量级：确保 TPA 执行审计验证的过程中，与 DSN 数据管理者进行交互的通信代价与计算开销要尽可能的小。

3.3 DSN 云存储数据完整性自我验证方案

3.3.1 方案使用的主要技术以及优势

基于椭圆曲线数字签名算法（ECDSA），本节提出了一个轻量级的 DSN 云存储数据自我完整性验证方案。方案支持数据的动态操作，与同类的工作^[26]相比，方案具有以下优势：

1. 本节方案验证过程只涉及到椭圆曲线上轻量级的乘法与加法计算。更进一步，在方案中 ECDSA 算法使用了更短的签名长度，需要更小的存储空间，这非常适用于无线通信以及一些在有条件约束下需要高效实现的环境。
2. 采用了一个轻量级的对称加密算法来加密 DSN 数据管理者的原始数据块，并将其发送给云服务器，因此数据块内容信息不会被云服务提供商获取。
3. 文献[26]中采用 Merkle 哈希树和 BLS 签名技术来实现数据的动态操作。我们的方案则是采用一个哈希索引列表[39]来支持数据的动态操作。通过哈希索引列表，DSN 数据管理者就能在每一个数据块上高效的执行动态操作，而不再需要重新计算其它的数据块。

3.3.2 系统框架与风险模型

根据之前的云存储数据完整性验证方案^[23-25]，一个自我验证的云存储数据完整性验证方案主要由以下四个算法构成：

1. $KeyGen(1^k)$: 算法输入一个安全参数 1^k , 输出一个公私钥对。
2. $SigGen(sk, F)$: 算法输入密钥 sk , 和数据文件 F , 输出一系列验证标签的集合 Φ 、文件名标识 t 。随后, DSN 数据管理者将 $\{F, \Phi, t\}$ 存储在云服务器。
3. $ProofGen(F, \Phi, chal)$: 算法由云服务器执行。挑战信息 $chal$ 由自我验证者生成。在这一步中, 算法输入挑战信息 $chal$, 文件 F , 相应的一系列验证标签集合 Φ , 输出一个相应的证明 P 。
4. $VerifyPoof(pk, chal, P)$: 一旦从云服务器收到相应证明 P , 算法输入 pk 、 $chal$ 和 P , 如果数据确实是完整的输出 $true$, 否则输出 $false$ 。

3.3.3 改进的 ECDSA

ECDSA (The Elliptic Curve Digital Signature Algorithm) 是一种类似数字签名算法 DSA (Digital Signature Algorithm) 的椭圆曲线签名算法。本节完整性验证方案通过改进 ECDSA 以继承其优势。修改后的 ECDSA 具有与标准 ECDSA 等同的安全性。

表 3-1 符号说明表

符号	含义
f	轻量级对称密码算法
$M = (m_1, m_2, \dots, m_n)$	含有 n 个数据块的数据文件
$t = id \parallel SSig_{ssk}(id)$	用户身份 id 的标签
$ECDSA$	椭圆曲线上的数字签名算法
$chal$	完整性验证挑战信息
G	生成元为 E 的循环群
(spk, ssk)	轻量级签名算法的公私钥对
$h : G \rightarrow Z_q$	将循环群中的元素映射到 Z_q 上的哈希函数
$K = \{q, C, E, F\} : F = xE$	椭圆曲线签名算法的公钥

1. 密钥生成步骤 $KeyGen$: 定义一个在 F_p 上的椭圆曲线 C , 其中 p 是一个大素数。设置 E 是椭圆曲线 C 上阶为素数 q 的一个点, E 是循环群 G 的生成元。设置 $M = \{0, 1\}^*$, 并计算 $F = xE$, 这里 $0 \leq x \leq q - 1$, 公钥为 $K = \{q, C, E, F\}$, 私钥为 x 。方案中使用安全的哈希函数 H (例如 $SHA-1$)。
2. 签名步骤 $Sign$: 对于给定消息 $m \in M$ 的签名如下: 秘密的选择随机数 ℓ , $0 \leq \ell \leq q - 1$, 计算 $Sig_x(m, \ell) = (r, s)$, 这里 $\ell E = (\alpha, \beta)$, $r = \alpha \bmod q$,

$s = (r\ell + H(m)x) \bmod q$ 。如果 $r = 0$ 或是 $s = 0$, 就选择另外的随机数 ℓ 。

3. 验证步骤 *Verify*: 一旦收到签名 $(m, (r, s))$, 验证者可通过以下步骤验证签名:

- (1) 计算 $w = r^{-1} \bmod q$;
- (2) 计算 $\lambda_1 = ws \bmod q$, $\lambda_2 = -wH(m) \bmod q$;
- (3) 计算 $(\alpha, \beta) = \lambda_1 E + \lambda_2 F$;
- (4) 当且仅当 $\alpha \bmod q = r$ 时验证 $ver_\ell(m, (r, s)) = true$ 是否成立。

由于 $(\alpha, \beta) = \lambda_1 E + \lambda_2 F = r^{-1}sE - r^{-1}H(m)F = r^{-1}sE - r^{-1}H(m)xF = r^{-1}(s - H(m)x)E$, 因此 $(\alpha, \beta) = \ell E$, $r = \alpha \bmod q$ 。所以, 验证过程满足正确性。更进一步, 如果验证标签能够被伪造, 那么它将同样解决循环群 G 上的离散对数问题, 因此改进后的 ECDSA 的安全性等价标准 ECDSA。

3.3.4 具体构造

在本节中, 基于改进椭圆曲线数字签名算法 (ECDSA), 提出了一个自我完整性验证方案, 方案的相关符号说明见表 3-1, 该方案由以下四个算法构成:

1. 系统初始化 *Setup*: 定义一个在 F_p 上的椭圆曲线 C , 其中 p 是一个大素数。设置 E 是椭圆曲线 C 上阶为素数 q 的一个点, E 是循环群 G 的生成元, 且 G 上的离散对数问题是难处理的。设置一个安全的对称加密算法 f , 其密钥为 τ 。DSN 数据管理者随机的选择 $x \leftarrow Z_q$, 并计算 $F = xE$, DSN 数据管理者的公钥和私钥分别为: $K = \{q, C, E, F\}$ 和 x 。具体的说, DSN 数据管理者使用一个随机签名密钥对 (spk, ssk) 来生成文件标识。密钥参数为: $sk = \{ssk, x, \tau\}$, 公共参数为: $pk = \{spk, q, E, F, C\}$ 。

2. 签名 *Sign*: 给定一个数据文件 $M = (m_1, m_2, \dots, m_n)$, id 是它的序号, 用来确保文件序号 id 的独立完整, DSN 数据管理者计算 $t = id \| SSig_{ssk}(id)$ 。然后, 管理者生成每一个数据块 m_i 的验证标签如下: 管理者先选择一个秘密的随机数 $\ell_i \in Z_q$, 并计算 $Q_i = \ell_i E = (\alpha_i, \beta_i)$, $r_i = \alpha_i \bmod q$, 以及 $s_i = (r_i \ell_i + m_i x) \bmod q$, 因此 DSN 数据管理者获得的验证标签为: $\sigma_i = (Q_i, r_i, s_i)$ ($i = 1, \dots, n$)。一系列的验证标签集合表示为 $\Phi = \{\sigma_i\}_{1 \leq i \leq n}$ 。同时为确保数据文件的机密性, DSN 数据管理者调用轻量级对称密码算法 f 使用对称密钥 τ 将每一个数据块 m_i 加密为 $m'_i = m_i + f_\tau(id \| i)$, 这样数据文件 $M = (m_1, m_2, \dots, m_n)$ 被加密为 $M' = (m'_1, m'_2, \dots, m'_n)$ 。最后 DSN 数据管理者将 $\{M', t, \Phi\}$ 发送给云服务器, 并同时将原来存储在本地存储系统中的数据文件 $M = (m_1, m_2, \dots, m_n)$ 、签名集合 $\Phi = \{\sigma_i\}_{1 \leq i \leq n}$ 和文件标识 t 删除。

3. 完整性验证证明生成*ProofGen*: DSN 数据管理者首先取回数据文件标识 t , 并用 spk 验证 $SSig_{ssk}(id)$, 如果验证失败, 则 DSN 数据管理者停止运行完整性验证。否则 DSN 数据管理者恢复 id 。

为生成完整性验证挑战信息, DSN 数据管理者在集合 $\{1, 2, \dots, n\}$ 中随机选取含有 c 个元素的子集合 \mathcal{J} ; 对于每一个 $j \in \mathcal{J}$, DSN 数据管理者产生一个相应的随机值 ν_j (比特长度应小于 q 的比特长度), 挑战数据块的位置信息应当被验证。随后, DSN 数据管理者向云服务器发送完整性验证挑战信息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ 。

一旦云服务器接收到完整性验证挑战信息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$, 执行 *ProofGen* 算法以生成证明响应信息如下: 云服务器计算 $s = \sum_{j \in \mathcal{J}} v_j s_j$, $Q = \sum_{j \in \mathcal{J}} v_j r_j Q_j$ 。进一步的, 云服务器计算抽样数据块的线性混合: $\mu' = \sum_{j \in \mathcal{J}} v_j m'_j$ 。然后云服务器将 $\{Q, s, \mu', id\}$ 作为存储正确性的证明响应发送给 DSN 数据管理者。

4. 完整性验证 *Verify*: 一旦收到云服务器的审计证明响应信息 $\{Q, s, \mu', id\}$, DSN 数据管理者调用轻量级的对称密码算法 f (对称密钥为 τ), 并计算 $\lambda = \mu' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j) \bmod q$, 然后利用 ECDSA 签名算法的公钥 $K = \{q, C, E, F\}$ 验证等式 $sE = Q + \lambda F$ 是否成立。如果等式成立, DSN 数据管理者就可以相信他存储在这个服务器上的数据块是完整的, 未被篡改。验证公式推导如下:

$$\begin{aligned}
 sE &= \sum_{j \in \mathcal{J}} v_j s_j E \\
 &= \sum_{j \in \mathcal{J}} v_j r_j \ell_j E + \sum_{j \in \mathcal{J}} v_j m_j x E \\
 &= \sum_{j \in \mathcal{J}} v_j r_j \ell_j E + \sum_{j \in \mathcal{J}} v_j m_j F \\
 &= \sum_{j \in \mathcal{J}} v_j r_j \ell_j E + (\mu' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j)) F \\
 &= Q + \lambda F
 \end{aligned} \tag{3-1}$$

3.3.5 安全性分析

定理 3.1: 在本节的自我完整性验证方案中恶意的云服务器如果能够生成可以通过验证等式的证明响应信息就能够解决 G 上的离散对数困难性问题。

证明: 在此方案中, 如果恶意云服务器可以赢得游戏 1、游戏 2 和游戏 3 中的任意一个, 它就可以生成一个伪造的云存储数据文件的证明响应信息, 从而成功的通过验证等式。现定义游戏 1, 2, 3 分别如下:

游戏 1: DSN 数据管理者发送一个完整性验证挑战信息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ 给云服务器, 则正确的证明响应信息应当为 $\{Q, s, \mu', id\}$, 可以通过验证等式: $sE = Q + (\mu' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j)) F$ 。恶意云服务器生成一个伪造的证明响应信息为 $\{Q, s, \mu'', id\}$ 给 DSN 数据管理者, 这里 $\mu'' = \sum_{j \in \mathcal{J}} v_j (m''_j + f_\tau(id \| j)) \neq \mu'$ 。如

果证明响应信息可以通过验证等式 $sE = Q + (\mu'' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j))F$, 则恶意云服务器赢得游戏, 否则失败。

现证明如果恶意云服务器可以赢得游戏 1, 我们就可以解决离散对数困难性问题, 这与事实相矛盾。首先我们假设恶意云服务器可赢得游戏 1, 根据上面的两个验证等式, 可得 $(\mu' - \mu'')F = 0$, 因此 $\sum_{j \in \mathcal{J}} v_j (m'_j - m''_j)F = \sum_{j \in \mathcal{J}} v_j \Delta m_j F = 0$ 。由于 G 是一个循环群, 对于群中的两个元素 $R, S \in G$, 存在 $u \in Z_q$ 使得 $S = uR$ 。不失一般性的给定 $R, S \in G$ 、 $v_j F = v_j x E$, 通过计算可生成 $v_j F = a_j R + b_j S$ 。因此 $\sum_{j \in \mathcal{J}} a_j \Delta m_j R + \sum_{j \in \mathcal{J}} b_j \Delta m_j S = 0$ 。由此, 可以解决 G 上的离散对数困难性问题。特别的如果下列等式分母部分不等于 0, 我们就可以从中得到 $S = -(\sum_{j \in \mathcal{J}} a_j \Delta m_j / \sum_{j \in \mathcal{J}} b_j \Delta m_j)R$ 。正如游戏 1 所描述, b_j 是 Z_q 中随机选择的元素, 因此在 $\Delta m_j (j \in \mathcal{J})$ 中至少存在一个元素不等于零, 因此我们得出使得分母为零的概率为 $1/q$ (可以忽略)。相应的, 如果恶意云服务器赢得游戏 1, 则解决离散对数困难性问题的概率为 $1 - 1/q$, 这是一个不可忽略的概率, 与事实相矛盾。

游戏 2: 如果恶意云服务器生成一个伪造的证明响应信息 $\{Q', s', \mu', id\}$, 其中 $\{Q', s'\} = \{Q, s\}$, 如果这个证明响应信息仍然能够通过验证等式 $s'E = Q' + (\mu' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j))F$, 则不可信云服务器赢得游戏。由上所知, 正确的完整性验证方案证明应当为 $\{Q, s, \mu', id\}$, 可以通过验证等式 $sE = Q + (\mu' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j))F$ 。根据两个等式, 可知 $(s - s')E = Q - Q'$, 由此可得 $Q = Q'$, 或者解决 E 和 U 之间的离散对数困难性问题 (这里有 $U = Q - Q'$)。这两个结果与我们的假设相矛盾。

游戏 3: 如果恶意云服务器生成一个伪造的证明响应信息 $\{Q', s', \mu'', id\}$, 若 $Q \neq Q'$, $s \neq s'$, $\mu'' \neq \mu'$, 这个伪造的完整性验证证明仍然能够通过验证等式 $s'E = Q' + (\mu'' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j))F$, 则不可信云服务器赢得游戏。这里, 定义 $\Delta s = s' - s$, $\Delta \mu = \mu'' - \mu'$ 。如上所知, 正确的完整性验证证明应当为 $\{Q, s, \mu', id\}$, 可以通过验证等式 $sE = Q + (\mu' - \sum_{j \in \mathcal{J}} v_j f_\tau(id \| j))F$ 。根据两个等式, 可知 $\Delta s E = Q' - Q + \Delta \mu F$, 由于 G 是一个循环群, 并且 $Q' - Q + \Delta \mu F \in G$, 不失一般性的设置 $V = Q' - Q + \Delta \mu F$, 所以 $\Delta s E = V$ 。因此可以解决 E 和 V 之间的离散对数困难性问题, 这就导致了矛盾。

因此, 对于一个恶意云服务器生成一个可以通过验证等式的伪造证明响应信息在计算上是不可行。■

注意: 如果 DSN 数据管理者想从一个数据文件的一些线性组合数据块中取得一个消息, 他只需要发送一个完整性验证挑战信息给云服务器, 并执行完整性验证过程。然后, DSN 数据管理者使用密钥解密这个从云服务器中得到的线性组合

加密文本，这样就可以取回预期的消息。

3.3.6 支持数据动态操作

在云存储系统中，出于各种需求^[29,40,42,104]，外存数据可能会被 DSN 数据管理者经常性的修改。因此，支持数据动态操作的云存储完整性验证方案在现实应用中是非常具有吸引力的。本节中提出的方案利用哈希索引列表^[39]来实现方案对数据动态操作的支持，具体支持的操作包括：修改、删除、插入。

本节方案中，一个数据文件的身份表示为 $id_j = \{c_j, h_j\}$ ，将其输入哈希表，这里 c_j 是数据文件 M'_j 的虚拟索引，使用一个安全的哈希函数 $H_1 : \{0, 1\} \rightarrow Z_q$ 得到 $h_j = H_1(M'_j || c_j)$ 。由于 H_1 的抗碰撞属性在虚拟目录中可以保证云存储所有数据文件的顺序是正确的，每一个文件在正确的顺序中都有一个独一无二的身份。动态操作的详细步骤如下：

数据修改：数据修改指将一个数据文件修改为新的数据文件。设 DSN 数据管理者欲将第 j 个数据文件修改为一个新的数据文件 M'_j 。则修改 h_j 为 h'_j , $h'_j = H_1(M'_j || c_j)$ 时，文件的虚拟索引保持不变。新数据文件身份索引修改为 $id_j = \{c_j, h'_j\}$ 。其它数据文件的身份保持不变。DSN 数据管理者为新的数据文件生成新的标签 $\{t'_j, \Phi'_j\}$ ，并上传 $\{t'_j, M'_j, \Phi'_j\}$ 到云服务器。

数据插入：当 DSN 数据管理者欲将一个新的数据文件 F'_j 插入到云存储数据中。DSN 数据管理者为文件 M'_j 计算对应的身份 $id_j = \{c'_j, h'_j\}$ 。虚拟索引则为 $c'_j = (c_{j-1} + c_j) / 2$, $h'_j = H_1(M'_j || c_j)$ 。其它的数据文件身份保持不变。DSN 数据管理者为插入部分 $\{id'_j, M'_j\}$ 生成新的标签 $\{t'_j, \Phi'_j\}$ ，并上传到云服务器。

数据删除：数据删除是数据插入的反向操作。DSN 数据管理者从云服务器删除数据文件 M'_j ，身份 id'_j 和标签 $\{t'_j, \Phi'_j\}$ 。其它数据文件的身份保持不变。

3.3.7 效率比较

在本节中，使用同类型的完整性验证方案^[26]与提出的轻量级的云存储完整性验证方案进行对比。由于在现实的应用环境中，云服务器拥有强大的计算能力，因此，下面只关注和讨论验证者在两个完整性验证方案中的计算开销与通信代价。本节对所涉及的计算量和通信量的符号说明如下：

$Mult_{G_1}, Mult_{G_T}, Mult_{Z_p}, Mult_{Z_q}, Mult_G$ 分别代表群 G_1 中一个乘法的计算量，双线性群 G_T 中一个乘法的计算量， Z_p, Z_q 中一个乘法的计算量，生成元为 E 的循环群 G 中一个乘法的计算量。

$Add_{Z_p}, Add_{Z_q}, Add_G$ 分别代表 Z_p, Z_q 中一个加法的计算量，循环群 G 中一个加

法的计算量。

$Hash_{Z_p}, Hash_{Z_q}, Hash_{G_1}$ 代表哈希值映射到 Z_p, Z_q 以及群 G_1 上需要的计算量。

$Exp_{G_1}, Exp_{G_T}, Pair_{G_1, G_T}, Enc_f$ 分别代表在群 G_1 和双线性群 G_T 中一个指数运算的计算量，一个双线性对的计算量，一个轻量级对称密码算法加密所需要的计算量。

$|p|, |q|$ 分别代表在 Z_p, Z_q 中的元素的比特数， $|G_1|, |G_T|, |G|$ 分别代表在群 G_1 、双线性群 G_T 、循环群 G 中一个元素的比特数， $|n|$ 代表数据块个数的比特长度。

1. 通信代价与计算开销：通信代价主要包含两个方面：完整性验证挑战消息与证明响应信息的通信量。对于每一个完整性验证挑战消息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ ，其通信代价为 $c(|n| + |p|)$ 。在完整性验证过程中云服务器生成的完整性验证证明为 $\{Q, s, \mu', id\}$ ，包含 G 中的两个元素， Z_q 中的一个元素，和 id ，其通信代价为 $2|G| + |q| + |id|$ 。因此，完整性验证的总共开销为 $2|G| + (c + 1)|q| + c|n| + |id|$ 。

计算开销在方案中主要集中于验证证明响应信息正确性的计算开销：即验证等式 $sE = Q + \lambda F$ 是否有效。

其计算开销为： $cEnc_f + cMult_{Z_q} + 3Mult_G + cAdd_{Z_q} + Add_G$ 。

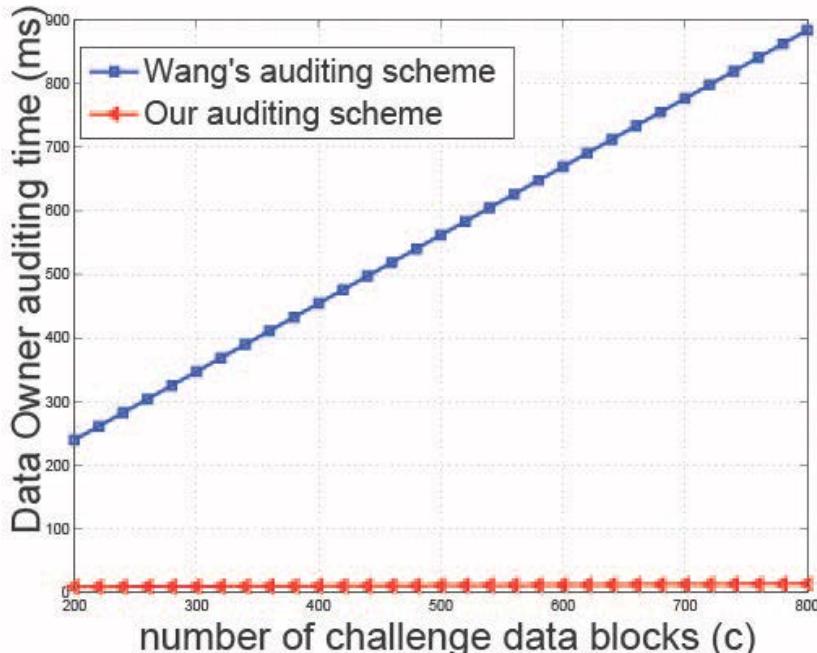


图 3-2 审计验证时间对比

2. 实验结果：实验平台如下：实验设备的操作系统为 windows 7，硬件配置是 Intel Core 2 i5 处理器，CPU 的频率为 2.53 GHz，内存为 2G DDR 3 of RAM(1.74 GB available)。本节使用的是 MIRACL 库，版本号为 5.6.1，开发环境为 Microsoft

Visual C 6.0。实验仿真所使用的椭圆曲线为 MNT 曲线，基本的文件大小为 159 比特，嵌入阶为 6。选择 $|p| = |q| = 160$ 比特，并简单化处理设置 $k = 20$, $c = 300$ 。

如图 3-2 所示，实验表明，本章完整性验证的计算开销要比方案[26]更为轻量级。具体来说，当挑战数据块的数量为 800 时，方案[26]的计算开销是本章方案的 40 倍。随着挑战数据块数量的增加，将会更高。这是因为方案[26]需要非常昂贵的对操作，这是需要消耗大量时间的。

3.4 DSN 隐私保护的云存储数据公开完整性验证方案

在 DSN 数据云储存系统中，DSN 用户能够享受到 DSN 与云存储带来的双重便利。但是，DSN 拥有者更多关注的还是如何满足传感器网络用户提出的各种需求，所以他希望一个第三方审计者来替他完成周期性的数据完整性验证任务。因此，本节提出了一种基于 DSN 应用的支持数据隐私保护的公开完整性验证方案。该方案同样不需要进行双线性对运算，有着较低的通信开销与计算代价。

3.4.1 方案具体构造

本节提出的 DSN 隐私保护的云存储数据公开完整性验证方案的验证过程如图 3-3 所示。这里，首先需要定义一个半可信的第三方审计者 TPA，这个第三方审计者 TPA 能够诚实地进行数据块完整性的审计验证工作。但同时，这个第三方审计者 TPA 也是充满好奇心的，它可能会尝试通过搜集审计验证过程中的验证信息来恢复 DSN 数据管理者的原始数据块。方案由四个子算法组成，分别是系统参数设置算法 $Setup$, 数据块签名生成算法 $SigGen$, 证明生成算法 $ProofGen$, 证明验证算法 $ProofVerify$, 具体描述如下：

1. 系统参数设置 $Setup$: 该算法负责初始化系统，选择两个大的素数 p 和 q ，其中 q 为满足 $p - 1$ 的素数因子；系统选择一个整数 g 满足 $g^q \equiv 1 \pmod{p}$; g 是阶为 q 的乘法循环群 G 的一个生成元。数据文件 M 被分成 n 块，每一个数据块被进一步分成 Z_q 中的 k 个元素。因此， M 可被表示为 $M = (m_1, m_2, \dots, m_n) \in Z_q^{n \times k}$, 对于任意一个 $m_j = (m_{j,1}, m_{j,2}, \dots, m_{j,k}) \in Z_q^k$, $1 \leq j \leq n$; 系统设置一个随机数生成器 $PRG : \mathcal{K}_{prg} \rightarrow Z_q^k$ 和一个伪随机函数 $PRF : \mathcal{K}_{prf} \times \mathcal{I} \rightarrow Z_q$, 这里的 \mathcal{K}_{prg} 和 \mathcal{K}_{prf} 分别表示 PRG 和 PRF 的秘密密钥集合, \mathcal{I} 表示数据文件 M 中所有数据块的身份集合。然后，DSN 数据管理者随机选择 $x \leftarrow Z_q$, 满足 $x \neq 0$, 并计算 $y = g^x \pmod{p}$ 。同时，DSN 数据管理者计算一个密钥对 $skp = (sk_{prg}, sk_{prf})$, 这里 $sk_{prg} \in \mathcal{K}_{prg}$, $sk_{prf} \in \mathcal{K}_{prf}$ 。系统设置一个轻量级的对称加密算法 f , 其私钥为 τ 。系统设置一个安全的哈希函数 $h : G \rightarrow Z_q$ 。需特别指明的是，为生成数据块验证标签，DSN 数

据管理者随机选择一组签名密钥对 (spk, ssk) 来进行数据块签名。因此公开参数为 $pk = \{G, g, y, spk\}$, 秘密参数为 $sk = \{x, \tau, ssk\}$ 。

2. 签名生成 $SigGen$: 给定一个数据块 $m_j = (m_{j,1}, \dots, m_{j,k})$, 标记数据块的身份为 $id_j \in \mathcal{I}$ 。为确保每一个数据块的身份标识都是唯一的, DSN 数据管理者计算 $tag_j = id_j \| SSig_{ssk}(id_j)$ 作为数据块 m_j 的标识。DSN 数据管理者计算 $\rho = (\rho_1, \dots, \rho_k) \leftarrow PRG(sk_{prg}) \in Z_q^k$ 和 $\omega_j \leftarrow PRF(sk_{prf}, id_j) \in Z_q$ 。然后, DSN 数据管理者为每一个数据块 $m_j = (m_{j,1}, \dots, m_{j,k})$ 计算一个同态消息认证码 $t_j = \sum_{\ell=1}^k \rho_\ell m_{j,\ell} + \omega_j \in Z_q$ 。DSN 数据管理者按照以下步骤计算 t_j 的签名:

- (1) 选择 $k_j \leftarrow Z_q$ 然后计算 $r_j = g^{k_j} \bmod p$ 和 $r'_j = r_j \bmod q$;
- (2) $s_j = (r'_j k_j + t_j x) \bmod q$;

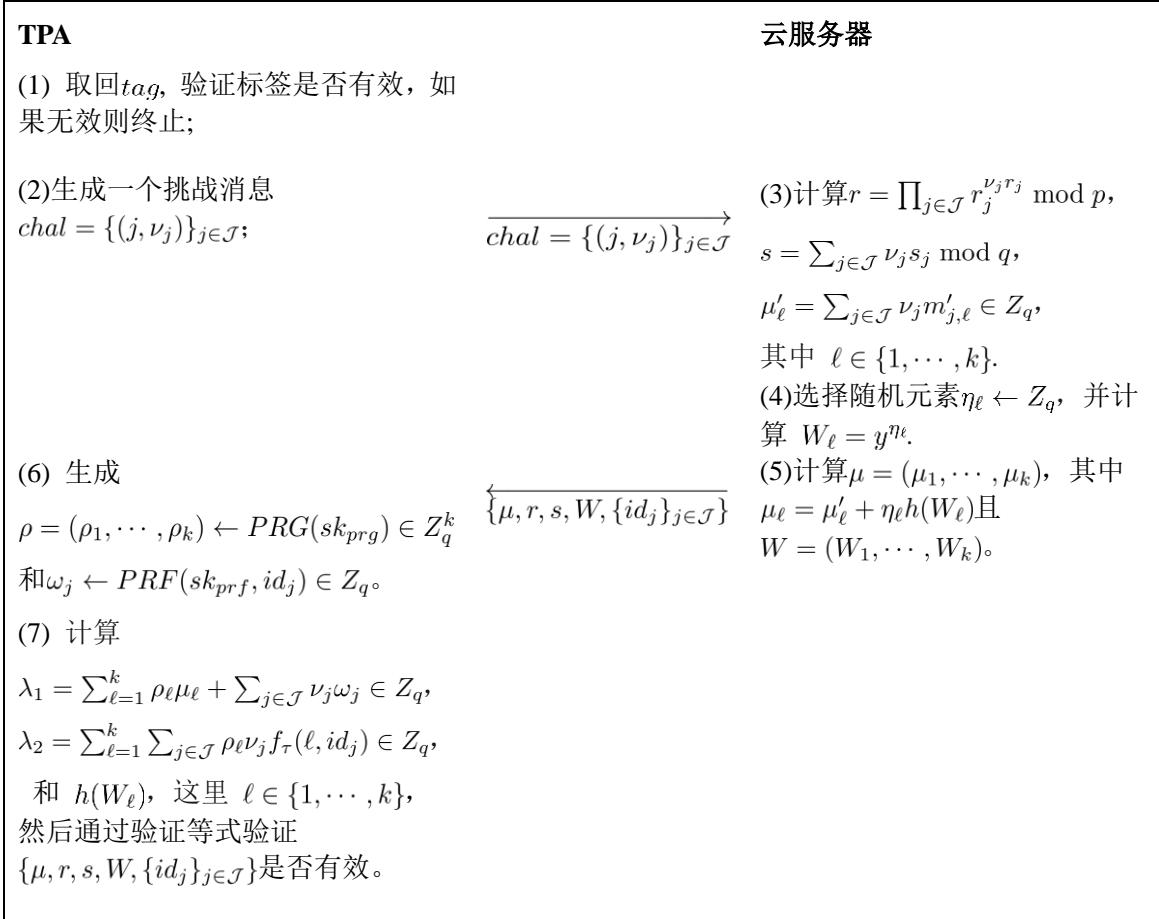


图 3-3 数据完整性验证过程

- (3) 输出 $\sigma_j = (r_j, s_j)$ 作为每一个同态消息认证码 t_j 的签名。

用 $\Phi = \{\sigma_j\}_{1 \leq j \leq n}$ 表示签名的集合。同时, 为保证数据文件的机密性, DSN 数据管理者使用一个轻量级的对称加密算法 f 来加密每一个数据块

$m_j = (m_{j,1}, \dots, m_{j,k})$ 为 $m'_j = (m_{j,1} + f_\tau(1, id_j), \dots, m_{j,k} + f_\tau(k, id_j))$, 其中加密密钥为 τ 。由此, 数据文件 $M = (m_1, \dots, m_n)$ 被加密成为 $M' = (m'_1, \dots, m'_n)$ 。最后, DSN 数据管理者发送 $\{M', tag_{1 \leq j \leq n}, \Phi\}$ 到云服务器, 并将其从本地删除。

3. 证明生成 $ProofGen$: 针对任意一个数据块 m_j , TPA 首先取回数据的标签, 再用 spk 验证这一块数据块对应的标识 $SSig_{ssk}(id_j)$ 。如果验证失败, 该算法中止; 否则, DSN 数据管理者恢复数据块身份 id_j 。

接下来, DSN 数据管理者为了验证数据文件的完整性, 首先发送一个完整性验证请求给 TPA。TPA 在收到这个完整性验证请求之后, 会按以下步骤生成一个完整性验证挑战消息:

- (1) TPA 从集合 $\{1, \dots, n\}$ 中随机选择一个 c 个元素的子集 \mathcal{J} , 这 c 个被选中的数据块将用在此次完整性验证挑战;
- (2) 针对每一个 $j \in \mathcal{J}$, TPA 选择一个对应的随机值 ν_j ;
- (3) TPA 输出一个完整性验证挑战信息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$, 并将这个挑战信息发送给云服务器; 这个挑战消息 $chal$ 指定到挑战完整性验证需要检验数据块的位置。

一旦云服务器在收到本次完整性验证挑战信息 $chal$ 后, 将按以下步骤生成一个审计证明响应信息, 来证明其拥有这些被选中的数据块, 并且这些数据块正确的存储在云服务器上:

- (1) 计算 $r = \prod_{j \in \mathcal{J}} r_j^{\nu_j r_j} \bmod p$;
- (2) 计算 $s = \sum_{j \in \mathcal{J}} \nu_j s_j \bmod q$;
- (3) 计算 μ'_ℓ 作为抽样数据块的线性混合: $\mu'_\ell = \sum_{j \in \mathcal{J}} \nu_j m'_{j,\ell} \in Z_q$, 这里有 $\ell \in \{1, \dots, k\}$ 。

为盲化 μ'_ℓ , 云服务器选择一个随机元素 $\eta_\ell \leftarrow Z_q$, 接着计算 $W_\ell = y^{\eta_\ell}$ 和 $\mu_\ell = \mu'_\ell + \eta_\ell h(W_\ell)$ 。然后, 云服务器发送 $\{\mu, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$ 给 TPA 用于完整性验证方案, 这里 $\mu = (\mu_1, \dots, \mu_k)$ 、 $W = (W_1, \dots, W_k)$ 。

4. 证明验证 $ProofVerify$: 给定一个审计证明响应信息 $\{\mu, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$ 和一个挑战消息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ 。TPA 验证证明响应信息有效性的步骤如下:

- (1) 生成 $\rho = (\rho_1, \dots, \rho_k) \leftarrow PRG(sk_{prg}) \in Z_q^k$ 和 $\omega_j \leftarrow PRF(sk_{prf}, id_j) \in Z_q$, $j \in \mathcal{J}$;
- (2) 计算 $\lambda_1 = \sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j \in \mathcal{J}} \nu_j \omega_j \in Z_q$, $\lambda_2 = \sum_{\ell=1}^k \sum_{j \in \mathcal{J}} \rho_\ell \nu_j f_\tau(\ell, id_j) \in Z_q$, 和 $h(W_\ell)$, 其中 $\ell \in \{1, \dots, k\}$;
- (3) TPA 通过验证等式 $g^s = ry^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ 是否成立来检查证明响应信息是否有效。

若验证式 $g^s = ry^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ 成立，则DSN数据管理者相信：存储在云服务器上的数据文件的完整性是可信的，数据文件没有被任何人修改；同时，由于随机掩码 $W_{\{1 \leq \ell \leq k\}}$ 的存在，TPA不可能从DSN数据管理者的数据文件中恢复出原始数据块。

3.4.2 方案安全性分析

本节将对所提方案进行安全性分析，分析包括正确性、不可伪造性与隐私保护性。另外考虑到方案的可扩展性，将其扩展支持批量完整性验证方案。

定理 3.2：方案满足正确性。

证明：根据完整性验证方案中的完整性验证过程，验证等式的正确性推导如下：

$$\begin{aligned}
 g^s &= g^{\sum_{j \in \mathcal{J}} \nu_j s_j} = g^{\sum_{j \in \mathcal{J}} \nu_j (r'_j k_j + t_j x \bmod q)} \bmod p \\
 &= g^{\sum_{j \in \mathcal{J}} \nu_j r'_j k_j} g^{\sum_{j \in \mathcal{J}} \nu_j t_j x} \bmod p \\
 &= \prod_{j \in \mathcal{J}} r_j^{\nu_j r_j} y^{\sum_{j \in \mathcal{J}} \nu_j t_j} \bmod p \\
 &= ry^{\sum_{j \in \mathcal{J}} (\sum_{\ell=1}^k \rho_\ell m_{j,\ell} + \omega_j) \nu_j} \bmod p \\
 &= ry^{\sum_{j \in \mathcal{J}} \sum_{\ell=1}^k \rho_\ell \nu_j m_{j,\ell} + \sum_{j \in \mathcal{J}} \omega_j \nu_j} \bmod p \\
 &= ry^{\sum_{\ell=1}^k \rho_\ell \sum_{j \in \mathcal{J}} \nu_j m_{j,\ell} + \sum_{j \in \mathcal{J}} \omega_j \nu_j} \bmod p \\
 &= ry^{\sum_{\ell=1}^k \rho_\ell (\mu_\ell - \sum_{j \in \mathcal{J}} \nu_j f_\tau(\ell, id_j) - \eta_\ell h(W_\ell)) + \sum_{j \in \mathcal{J}} \nu_j \omega_j} \bmod p \\
 &= ry^{\sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j \in \mathcal{J}} \nu_j \omega_j} y^{-\sum_{\ell=1}^k \sum_{j \in \mathcal{J}} \rho_\ell \nu_j f_\tau(\ell, id_j)} y^{-\sum_{\ell=1}^k \rho_\ell \eta_\ell h(W_\ell)} \bmod p \\
 &= ry^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p
 \end{aligned} \tag{3-2}$$

得出验证等式 $g^s = ry^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ 成立。

经上证明，完整性验证方案是正确的。■

定理 3.3：方案满足不可伪造性。恶意的云服务器产生伪造的审计证明响应信息 $Proof$ 来欺骗TPA通过完整性验证的过程在计算上是不可行的。

证明：在本章所提出的完整性验证中，使用同态消息认证码来压缩每一个数据块以显著地减少存储验证信息所需要的存储空间。根据文献[75]中的讨论与证明，可以得知一个敌手攻破一个数据块对应的同态消息认证码的概率为 $1/q$ ，显然，这个概率是可以忽略的，在此我们就不再累述。

除了伪造一个同态消息认证码外，如果恶意云服务器能够赢得下面的游戏，它就能够针对挑战数据块生成一个伪造的审计证明响应信息，并确保这个伪造的审计证明响应信息可以成功的通过验证。现描述游戏如下：

游戏：一旦收到一个来自DSN数据管理者的完整性验证请求，TPA发送一个完整性验证挑战信息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ 给云服务器。作为云服务器收到挑战验证

信息后生成的正确审计证明响应信息应当为 $\{\mu, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$, 其中 $\mu = (\mu_1, \dots, \mu_k), W = (W_1, \dots, W_k)$ 。显然, 这个证明响应信息是能够通过验证等式检验的。现为取代正确的审计证明响应信息, 恶意云服务器基于篡改的数据文件 M'^* 生成一个伪造的审计证明响应信息 $\{\mu^*, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$, 其中有 $\mu^* = (\mu_1^*, \dots, \mu_k^*), \mu_\ell^* = \mu_\ell^* + \eta_\ell h(W_\ell), \mu_\ell'^* = \sum_{j \in \mathcal{J}} \nu_j m_{j,\ell}^* \in Z_q$ 。设 $\Delta\mu_\ell = \mu_\ell^* - \mu_\ell$, 其中 $1 \leq \ell \leq k$ 。由于 $M' \neq M'^*$ 这里最少有一个元素 $\{\Delta\mu_\ell\}_{1 \leq \ell \leq k}$ 是非零知识的。如果这个伪造的完整性响应证明信息仍然能通过验证, 恶意的云服务器就赢得了这个游戏。否则失败。

接下来, 展示如果恶意云服务器赢得游戏, 就能找到一种方法解决离散对数困难性问题。首先假设这个恶意的云服务器能赢得游戏。那么根据验证等式, 能得到 $g^s = ry^{\lambda_1^* - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \pmod{p}$, 其中 $\lambda_1^* = \sum_{\ell=1}^k \rho_\ell \mu_\ell^* + \sum_{j \in \mathcal{J}} \nu_j \omega_j \in Z_q$ 。由于 $\{\mu, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$ 是正确的审计证明响应信息, 也可得到 $g^s = ry^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \pmod{p}$ 。再根据这两个认证等式, 我们能得到 $y^{\lambda_1^*} = y^{\lambda_1}$ 。因此有:

$$\begin{aligned} y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell^* + \sum_{j \in \mathcal{J}} \nu_j \omega_j} &= y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j \in \mathcal{J}} \nu_j \omega_j} \\ y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell^*} &= y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell} \\ y^{\sum_{\ell=1}^k \rho_\ell \Delta\mu_\ell} &= \prod_{\ell=1}^k (y^{\rho_\ell})^{\Delta\mu_\ell} = 1 \end{aligned} \quad (3-3)$$

由于 G 是一个 q 阶乘法循环群, 对两个随机元素 $\alpha, \beta \in G$, 存在 $\eta \in Z_q$ 使得 $\beta = \alpha^\eta$ 。不失一般性的给定 $\alpha, \beta \in G$, 每一个 y^{ρ_ℓ} 都是通过选择 Z_q 上的随机值 ξ_ℓ 和 γ_ℓ 计算生成 $y^{\rho_\ell} = \alpha^{\xi_\ell} \beta^{\gamma_\ell}$ 。然后可得:

$$\begin{aligned} 1 &= \prod_{\ell=1}^k (y^{\rho_\ell})^{\Delta\mu_\ell} \\ &= \prod_{\ell=1}^k (\alpha^{\xi_\ell} \beta^{\gamma_\ell})^{\Delta\mu_\ell} \\ &= \alpha^{\sum_{\ell=1}^k \xi_\ell \Delta\mu_\ell} \cdot \beta^{\sum_{\ell=1}^k \gamma_\ell \Delta\mu_\ell} \end{aligned} \quad (3-4)$$

显然, 通过上式所示可以找到一个方法来求解离散对数困难性问题。尤其是, 给定 $\alpha, \beta = \alpha^\eta \in G$, 能够输出 $\beta = \alpha^\eta = \alpha^{-\sum_{\ell=1}^k \xi_\ell \Delta\mu_\ell / \sum_{\ell=1}^k \gamma_\ell \Delta\mu_\ell}$, 其中 γ_ℓ 是 Z_q 上的一个随机元素; 由此得到 $\eta = -\sum_{\ell=1}^k \xi_\ell \Delta\mu_\ell / \sum_{\ell=1}^k \gamma_\ell \Delta\mu_\ell$, 除非这个分母是零。否则, 正如我们在上述游戏中定义的那样, 这里至少有一个元素 $\{\Delta\mu_\ell\}$ 是非零的。因此, 分母是零的概率为 $1/q$, 这个概率是可以忽略的。这意味着, 一旦恶意云服务器赢得了这个攻击游戏, 就能以不可忽略的概率 $1 - 1/q$ 找到一个方法解决离散对数计算困难问题。这就与离散对数在 G 上是计算不可行的假设相矛盾。

进一步, 如果恶意云服务器尝试伪造聚合签名, 就意味着, 如果云服务器生成了一个仍然能够通过验证等式 $g^{s'} = r'y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \pmod{p}$ 的伪造的审

计证明响应信息 $\{\mu, r', s', W, \{id_j\}_{j \in J}\}$, 就代表恶意服务器攻击成功。如上文所述, 可以通过验证等式 $g^s = ry^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ 的正确审计证明响应信息为 $\{\mu, r, s, W, \{id\}_{j \in J}\}$ 。根据这两个验证等式能够得到 $g^{s'-s} = r'r^{-1} \bmod p$, 由此可以得到 $s = s'$ 和 $r = r'$, 否则就能基于 g 和 d (这里设置 $d = r'r^{-1}$)找到一个方法来解决离散对数困难性问题, 这就导致了矛盾。

因此, 对于一个恶意云服务器生成一个伪造的审计证明响应信息, 以通过验证等式, 这在计算上是不可行的。■

定理 3.4: 给定一个来自于云服务器的审计证明响应信息 $proof = \{\mu, r, s, W, \{id_j\}_{j \in J}\}$, 对于好奇的 TPA, 它试图从中获取 DSN 数据管理者数据文件中原始数据块的任何隐私信息在计算上是不可行的。

证明: 如果 $\mu'_\ell = \sum_{j \in J} \nu_j m'_{j,\ell} \in Z_q$ 是关于数据块的线性组合, 一旦这个组合信息发送给 TPA, 这个好奇的 TPA 可以通过收集大量的组合信息, 并借助强大的计算设备来求解这些线性方程组, 从而恢复用户的原始数据块。为了防止 TPA 读取用户的原始数据块信息, $\mu'_\ell = \sum_{j \in J} \nu_j m'_{j,\ell} \in Z_q$ 需要使用随机掩饰码技术, 盲化成 $\mu_\ell = \mu'_\ell + \eta_\ell h(W_\ell)$ 。为了让 TPA 仍能解这些线性方程, TPA 必须知道 η_ℓ 的值, 然而给定 y , $W_\ell = y^{\eta_\ell} \in G$, 计算 η_ℓ , 是与在 G 上计算解决离散对数困难性问题等价的, 而解决离散对数困难性问题在计算上不可行的。因此, 给定审计证明响应信息, TPA 不能直接从数据块的线性混合中获取数据块元素, 更不能依靠解线性方程进一步恢复出有关数据文件的任何原始数据块。■

3.4.3 支持批处理的扩展方案

随着隐私保护数据完整性验证方案在云存储上的应用, TPA 可能在较短的时间内收到从不同的用户发送来的大量的审计请求。不幸的是, 只允许 TPA 执行分离的审计验证任务是一个乏味的并且非常低效的过程。因此, 进一步扩展本文数据完整性验证方案使其支持批量审计验证任务就显得非常重要。批量审计不仅能使 TPA 在同时执行多审计验证任务, 并显著减少 TPA 的计算代价。因为聚合 L 个等式一起进行验证, 能节约一个可预期的大量审计时间^[103]。细节描述如下:

- 系统初始化阶段:** DSN 数据管理者独立执行设置阶段, 假设完整性验证系统里有 L 个 DSN 数据管理者, 任意一个 DSN 数据管理者 θ 的一个数据文件 $M_\theta = \{m_{\theta,1}, \dots, m_{\theta,n}\}$ 被外包存储在云服务器上, 这里有 $m_{\theta,j} = (m_{\theta,j,1}, \dots, m_{\theta,j,k})$, $j = 1, 2, \dots, n$ 。出于简化处理, 假设每一个数据文件 M_θ 都有相同数量的数据块计为 n 。特别的, 对于任意一 DSN 数据管理者 θ , 他的私有参数表示为 $(x_\theta, ssk_\theta, sk_{prg_\theta}, sk_{prf_\theta})$, 对应的公开参数为 $(G, g, y_\theta, spk_\theta)$, 其中 $y_\theta = g^{x_\theta}$ 。类似于单

一 DSN 数据管理者方案，每一个 DSN 数据管理者 θ 为每一个数据块 $m_{\theta,j}$ 随机选择一个不同的身份标识 $id_{\theta,j}$ ，同时正确的生成相应的数据块身份标识 $tag_{\theta,j} = id_{\theta,j} \| SSig_{ssk_\theta}(id_{\theta,j})$ 。

然后，任意一 DSN 数据管理者 θ 计算 $\rho_\theta = (\rho_{\theta,1}, \dots, \rho_{\theta,k}) \leftarrow PRG(sk_{prg_\theta}) \in Z_q^k$ 和 $\omega_{\theta,j} \leftarrow PRF(sk_{prf_\theta}, id_{\theta,j}) \in Z_q$ 。接着，DSN 数据管理者为每一个数据块 $m_{\theta,j} = (m_{\theta,j,1}, \dots, m_{\theta,j,k})$ 计算同态消息认证码 $t_{\theta,j} = \sum_{\ell=1}^k \rho_{\theta,\ell} m_{\theta,j,\ell} + \omega_{\theta,j} \in Z_q$ 。DSN 数据管理者按如下步骤计算标签 $t_{\theta,j}$:

- (1) 选择 $k_{\theta,j} \leftarrow Z_q$ 然后计算 $r_{\theta,j} \equiv g^{k_{\theta,j}} \pmod{p}$ 和 $r'_{\theta,j} \equiv r_{\theta,j} \pmod{q}$;
- (2) $s_{\theta,j} = (r'_{\theta,j} k_{\theta,j} + t_{\theta,j} x) \pmod{q}$;
- (3) 输出 $\sigma_{\theta,j} = (r_{\theta,j}, s_{\theta,j})$ 作为每一个同态消息认证码 $t_{\theta,j}$ 的签名。

用 $\Phi_\theta = \{\sigma_{\theta,j}\}_{1 \leq j \leq n}$ 表示签名的集合。同时，为保证数据文件的机密性，DSN 数据管理者利用轻量级的对称加密算法 f 来加密每一个数据块 $m_{\theta,j} = (m_{\theta,j,1}, \dots, m_{\theta,j,k})$ 为 $m'_{\theta,j} = (m_{\theta,j,1} + f_{\tau_\theta}(1, id_{\theta,j}), \dots, m_{\theta,j,k} + f_{\tau_\theta}(k, id_{\theta,j}))$ ，其中加密密钥为 τ_θ 。由此，数据块文件 $M_\theta = (m_{\theta,1}, \dots, m_{\theta,n})$ 被加密为 $M'_\theta = (m'_{\theta,1}, \dots, m'_{\theta,n})$ 。最后，DSN 数据管理者 θ 发送 $\{M'_\theta, \{tag_{\theta,j}\}_{1 \leq j \leq n}, \Phi_\theta\}$ 给云服务器，再将其从本地存储中删除。

2. 审计阶段：TPA 首先为每一个 DSN 数据管理者 θ 取回并验证数据块标识 $tag_{\theta,j}$ 。如果验证失败，TPA 终止执行。否则，TPA 恢复数据块身份信息 $id_{j,\theta}$ 并将完整性验证挑战消息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ 发送给云服务器。同时，针对每一个 DSN 数据管理者 θ ，云服务器随机选择一个 $\eta_{\theta,\ell} \in Z_q$ 并计算 $W_{\theta,\ell} = y_\theta^{\eta_{\theta,\ell}}$ 和 $\mu_{\theta,\ell} = \sum_{j \in \mathcal{J}} \nu_j m'_{\theta,j,\ell} + \eta_{\theta,\ell} h(W_{\theta,\ell})$ 。由此，云服务器可得 $\mu_\theta = (\mu_{\theta,1}, \dots, \mu_{\theta,\ell}, \dots, \mu_{\theta,k})$ 。接着，云服务器完成聚合 $r = \prod_{\theta=1}^L \prod_{j \in \mathcal{J}} r_{\theta,j}^{\nu_j r_{\theta,j}}$ mod p 与 $s = \sum_{\theta=1}^L \sum_{j \in \mathcal{J}} \nu_j s_{\theta,j} \pmod{q}$ 。最后，云服务器发送响应消息为： $(\{\mu_\theta\}_{1 \leq \theta \leq L}, r, s, \{W_\theta\}_{1 \leq \theta \leq L}, \{id_{\theta,j}\}_{j \in \mathcal{J}, 1 \leq \theta \leq L})$ ，其中 $W_\theta = (W_{\theta,1}, \dots, W_{\theta,\ell}, \dots, W_{\theta,k})$ 。

为验证这个完整性证明响应消息，TPA 工作如下：

- (1) 生成 $\rho_\theta = (\rho_{\theta,1}, \dots, \rho_{\theta,k}) \leftarrow PRG(sk_{prg_\theta}) \in Z_q^k$, $\omega_{\theta,j} \leftarrow PRF(sk_{prf_\theta}, id_{\theta,j}) \in Z_q$, $j \in \mathcal{J}$;
- (2) 计算 $\lambda_{\theta,1} = \sum_{\ell=1}^k \rho_{\theta,\ell} \mu_{\theta,\ell} + \sum_{j \in \mathcal{J}} \nu_j \omega_{\theta,j} \in Z_q$, $\lambda_{\theta,2} = \sum_{\ell=1}^k \sum_{j \in \mathcal{J}} \rho_{\theta,\ell} \nu_j f_{\tau_\theta}(\ell, id_{\theta,j}) \in Z_q$ 和 $h(W_{\theta,\ell})$ ，其中 $1 \leq \ell \leq k$ 和 $1 \leq \theta \leq L$ 。

接着 TPA 验证认证等式 $g^s = r \prod_{\theta=1}^L y_\theta^{\lambda_{\theta,1} - \lambda_{\theta,2}} (\prod_{\ell=1}^k W_{\theta,\ell}^{-\rho_{\theta,\ell} h(W_{\theta,\ell})}) \pmod{p}$ 是否成立。验证等式正确性的推导过程如下：

$$\begin{aligned} g^s &= g^{\sum_{\theta=1}^L \sum_{j \in \mathcal{J}} \nu_j s_{\theta,j}} \pmod{p} \\ &= \prod_{\theta=1}^L g^{\sum_{j \in \mathcal{J}} \nu_j s_{\theta,j}} \pmod{p} \\ &= \prod_{\theta=1}^L g^{\sum_{j \in \mathcal{J}} \nu_j (r'_{\theta,j} k_{\theta,j} + t_{\theta,j} x_\theta \pmod{q})} \pmod{p} \end{aligned}$$

$$\begin{aligned}
 &= \prod_{\theta=1}^L g^{\sum_{j \in \mathcal{J}} \nu_j r'_{\theta,j} k_{\theta,j}} g^{\sum_{j \in \mathcal{J}} \nu_j t_{\theta,j} x_{\theta}} \bmod p \\
 &= \prod_{\theta=1}^L \prod_{j \in \mathcal{J}} r_{\theta,j}^{\nu_j r_{\theta,j}} y_{\theta}^{\sum_{j \in \mathcal{J}} \nu_j t_{\theta,j}} \bmod p \\
 &= r \prod_{\theta=1}^L y_{\theta}^{\sum_{j \in \mathcal{J}} \nu_j (\sum_{\ell=1}^k \rho_{\theta,\ell} m_{\theta,j,\ell} + \omega_{\theta,j})} \bmod p \\
 &= r \prod_{\theta=1}^L y_{\theta}^{\sum_{\ell=1}^k \rho_{\theta,\ell} \sum_{j \in \mathcal{J}} \nu_j m_{\theta,j,\ell} + \sum_{j \in \mathcal{J}} \nu_j \omega_{\theta,j}} \bmod p \\
 &= r \prod_{\theta=1}^L y_{\theta}^{\sum_{\ell=1}^k \rho_{\theta,\ell} (\mu_{\theta,\ell} - \sum_{j \in \mathcal{J}} \nu_j f_{\tau_{\theta}}(\ell, id_{\theta,j}) - \eta_{\theta,\ell} h(W_{\theta,\ell})) + \sum_{j \in \mathcal{J}} \nu_j \omega_{\theta,j}} \bmod p \\
 &= r \prod_{\theta=1}^L y_{\theta}^{\sum_{\ell=1}^k \rho_{\theta,\ell} \mu_{\theta,\ell} + \sum_{j \in \mathcal{J}} \nu_j \omega_{\theta,j} - \sum_{\ell=1}^k \sum_{j \in \mathcal{J}} \rho_{\theta,\ell} \nu_j f_{\tau_{\theta}}(\ell, id_{\theta,j}) - \sum_{\ell=1}^k \rho_{\theta,\ell} \eta_{\theta,\ell} h(W_{\theta,\ell})} \bmod p \\
 &= r \prod_{\theta=1}^L y_{\theta}^{\lambda_{\theta,1} - \lambda_{\theta,2}} y_{\theta}^{-\sum_{\ell=1}^k \rho_{\theta,\ell} \eta_{\theta,\ell} h(W_{\theta,\ell})} \bmod p \\
 &= r \prod_{\theta=1}^L y_{\theta}^{\lambda_{\theta,1} - \lambda_{\theta,2}} (\prod_{\ell=1}^k W_{\theta,\ell}^{-\rho_{\theta,\ell} h(W_{\theta,\ell})}) \bmod p
 \end{aligned} \tag{3-5}$$

由此，可知验证等式 $g^s = r \prod_{\theta=1}^L y_{\theta}^{\lambda_{\theta,1} - \lambda_{\theta,2}} (\prod_{\ell=1}^k W_{\theta,\ell}^{-\rho_{\theta,\ell} h(W_{\theta,\ell})}) \bmod p$ 成立。

3.4.4 方案的效率比较

在本节，将所提出的高效的隐私保护云存储数据公开完整性验证方案与文献[26]中的方案进行效率对比。首先对方案的计算开销与通信代价进行分析。接着，利用实验对这两个方案进行评估，并对执行效率做出对比，结果表明本章方案具有效率优势。

1. 计算开销分析：对高效的隐私保护的云存储数据公开完整性验证方案的计算开销与文献[26]中完整性验证方案的计算开销进行对比。本节方案中主要的密码学操作包括乘法、加法和哈希函数操作。为简化表示，忽略了伪随机数生成PRG和伪随机函数PRF生成操作，因为他们相对于前面提到的三种计算操作要简单和快速得多。这里使用 $Mult_G$, Add_G , 和 Exp_G 来分别表示群 G 上的乘法运算、加法运算、模指数运算操作的计算开销，同样使用 $Hash_G$ 来表示群 G 上的哈希函数运算开销，再用 $pair_{G_1, G_2}$ 来表示双线性对操作的计算开销。

在完整性验证过程中，TPA 首先生成一些随机值来构建完整性验证挑战消息，这个过程中只引入了少量的计算开销。接着，云服务器在收到完整性验证挑战消息后，需要计算一个证明 $\{\mu, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$ 来完成 TPA 的完整性验证方案，其中 $\mu = (\mu_1, \dots, \mu_k)$, $W = (W_1, \dots, W_k)$ 。这个证明的计算代价为：
 $(kc + 2c + k)Mult_{Z_p} + cMult_{Z_q} + kcAdd_{Z_p} + (c - 1)Add_{Z_q} + kExp_{Z_p} + kHash_{Z_p}$ ，
同时对文献[26]进行分析，文献[26]中方案生成证明的计算代价为：
 $(c - 1)Mult_{G_1} + (c + 1)Mult_{Z_p} + cExp_{G_1} + Exp_{G_T} + cAdd_{Z_p} + Hash_{Z_p}$ 。

为验证证明的有效性，TPA 通过验证等式验证完整性验证证明的开销为：

$$(2k + c + 2ck)Mult_{Z_p} + kMult_{Z_p} + (k + 2)Exp_{Z_p} + (ck + c + k - 2)Add_{Z_q} + 2kHash_{Z_q} + cEnc_{\varepsilon}$$

同时文献[26]中方案验证完整性验证证明的计算代价为：

$$(c + 1)Mult_{G_1} + Mult_{G_T} + (c + 3)Exp_{G_1} + 2Pair_{G_1, G_2} + Hash_{Z_p} + cHash_{G_1}$$

2. 通信代价分析：根据方案描述，通信代价主要来自于两个方面：一是完整性验证挑战消息的传输；二是审计证明响应信息的传输。对于挑战消息 $chal = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ ，云服务器生成的审计证明响应信息为 $\{\mu, r, s, W, \{id_j\}_{j \in \mathcal{J}}\}$ ，其中 $\mu = (\mu_1, \dots, \mu_k)$ 和 $W = (W_1, \dots, W_k)$ 。因此，方案总共的通信代价为 $(c + k + 1)|q| + c|n| + (k + 1)|p|$ 。而文献[26]中的方案总共的通信代价为 $c(|p| + |n|) + |p| + |G_1| + |G_T| + |id|$ ，其中 $|n|$ 是代表数据块个数的比特长度而 $|G_T|$ 是群 G_T 中一个元素的长度。因此，本章完整性验证方案对比文献[26]中方案在通信代价与计算开销总体上具有优势。

3. 实验分析

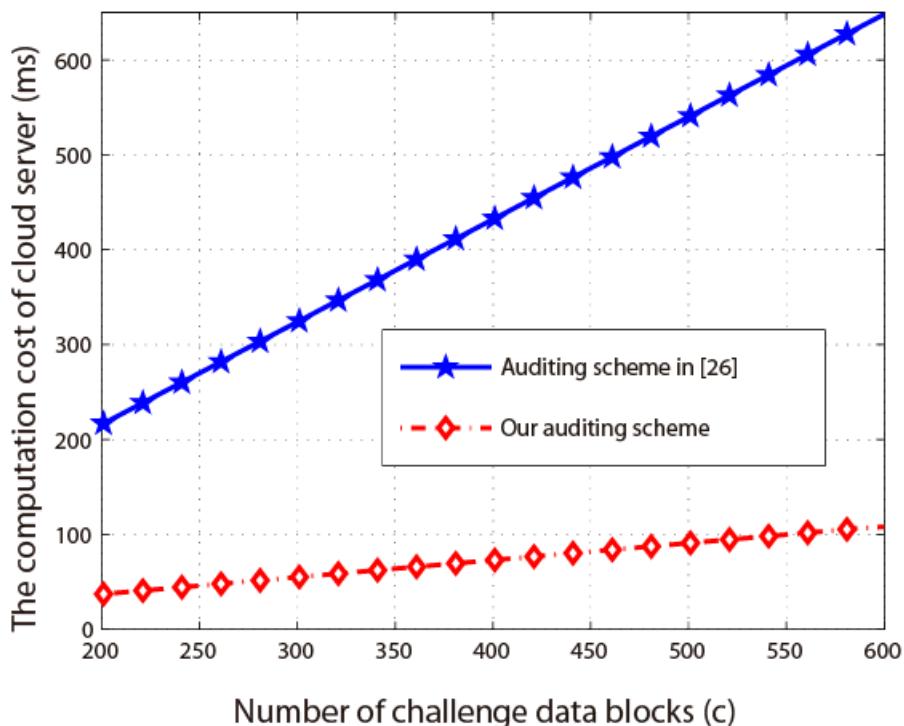


图 3-4 云服务器计算开销对比

现根据实验对本章完整性验证方案与文献[26]方案中云服务器的计算开销与TPA的完整性验证计算开销进行对比。由于随机盲化需要一个指数操作、一个乘法运算、一个哈希函数运算和一个加法运算。所以针对随机模化造成的额外计算开销的总和是一个常量： $Exp_{G_T} + Mult_{Z_p} + Hash_{Z_p} + Add_{Z_p}$ ，与抽样数据块 c 的数量无关。为确保抽样审计的正确性， c 的集合应设为 400 到 600。由于云服务器方面为数据隐私保护所需要的额外开销相对于服务器证明响应信息生成的总共计算开销是可以忽略的。因此，实验中，文献[26]中方案的云服务器的主要计算开销

为 $(c - 1)Mult_{G_1} + cMult_{Z_p} + cExp_{G_1} + (c - 1)Add_{Z_p}$ 。而在本文的完整性验证方案中，由于 k 是一个比抽样挑战数据块 c 要小得多的值，所以随机盲化造成的额外计算开销也是一个很小的常量值 $k(Exp_{Z_p} + Mult_{Z_p} + Hash_{Z_q} + Add_{Z_q})$ ，考虑到模指数运算操作计算开销要远大于其他操作，所以 $k(Mult_{Z_p} + Hash_{Z_q} + Add_{Z_q})$ 部分计算开销在实验中被忽略。

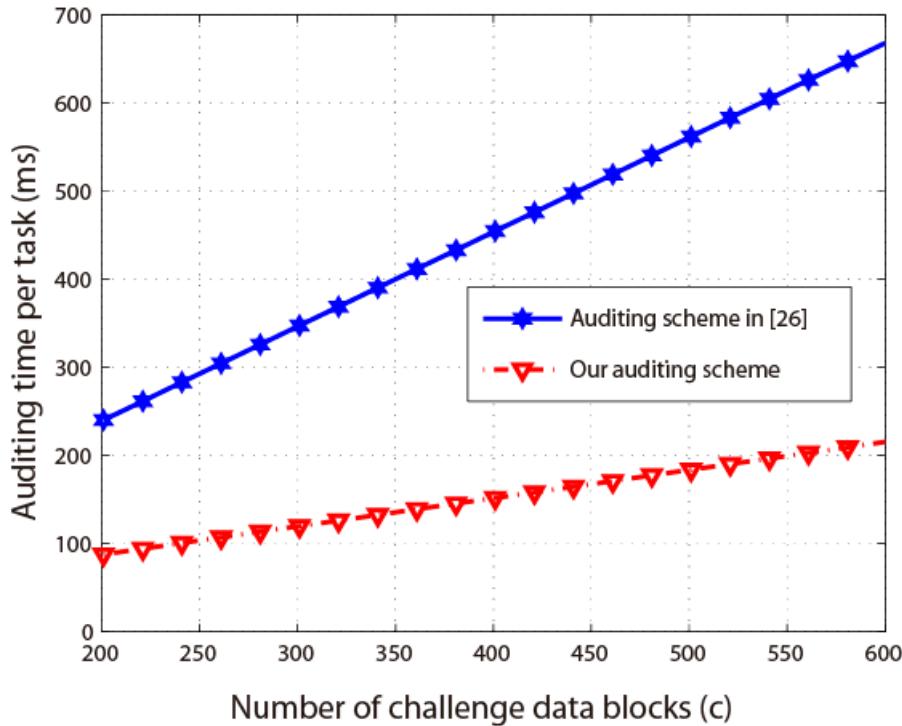


图 3-5 TPA 验证时间对比

将 $(kc + 2c)Mult_{Z_p} + cMult_{Z_q} + (kc - k)Add_{Z_p} + (c - 1)Add_{Z_q} + kExp_{Z_p}$ 设置为实验中本节所提方案云服务器的主要计算开销。并且设置文献[26]中方案的TPA的主要完整性验证计算开销为 $cMult_{G_1} + (c + 1)Exp_{G_1} + 2Pair_{G_1, G_2} + cHash_{G_1}$ 。本节方案中，出于一致性考虑，忽略掉计算开销 $k(Mult_{Z_p} + Mult_{Z_q} + Hash_{Z_q})$ ， $(k + c + 2ck)Mult_{Z_q} + (k + 2)Exp_{Z_p} + (ck + c + k - 2)Add_{Z_q} + kHash_{Z_q} + cEnc_\varepsilon$ 被设定为所提方案中TPA完整性验证过程的主要计算开销。

实验平台如下：实验设备的操作系统为 windows 7，硬件配置是 Intel Core 2 i5 处理器，CPU 的频率为 2.53 GHz，内存为 2G DDR 3 of RAM(1.74 GB available)。本节使用的是 MIRACL 库，版本号为 5.6.1，开发环境为 Microsoft Visual C 6.0。实验仿真所使用的椭圆曲线为 MNT 曲线，基本的文件大小为 159 比特，嵌入阶为 6。选择 $|p| = |q| = 160$ 比特，并简单化处理设置 $k = 20$, $c = 300$ 。

实验结果如图 3-4、3-5 所描述，相比文献[26]中的方案，本章完整性验证方案

中云服务器的计算开销与 TPA 的完整性验证时间开销都更为轻量级。尤其是，随着挑战数据块规模的增加，本章完整性验证方案将比文献[26]中方案在计算开销上具有更多的优势。造成这个优势的主要原因是（1）文献[26]中方案的云服务器计算开销中 $cExp_{G_1}$ 要远大于本节提出方案相对应的部分 $kExp_{Z_p}$ 。（2）文献[26]中方案的 TPA 完整性验证计算开销需要进行时间代价昂贵的双线性对运算，而所提方案不需要进行双线性对运算。

3.5 本章小结

本章主要研究适用于 DSN 的高效云存储数据完整性验证方案。具体包括：基于椭圆曲线数据签名算法（ECDSA）提出的轻量级的云存储数据自我完整性验证方案；与利用变型的 Schnorr 签名算法和同态消息认证码技术构建的高效的支持隐私保护的云存储数据公开完整性验证方案。两个方案的验证过程均不存在双线性对操作，并分别支持数据的动态操作与批量任务处理。通过对方案的安全性进行分析，证明方案能够有效缓解 DSN 数据管理者对外存数据安全威胁的担忧。效率比较与实验结果表明本章方案由于不涉及到代价昂贵的双线性对运算，所以方案是轻量级的，适用于分布式传感器网络应用场景。

第四章 支持用户可撤销的云存储数据完整性验证方案

本章研究的主要内容是支持用户可撤销系统中云存储数据的完整性验证问题。随着大数据时代的到来，由于可以享受高质量的云存储服务，公司和组织机构越来越倾向于将数据存储于云上。出于安全性的考虑，这些公司和组织往往需要对他们存储于云上的数据进行完整性检验。要做到这一点，他们需要一个恰当的符合实际需求的云存储数据完整性验证方案。但目前的研究往往集中于当前数据管理者的数据完整性验证需求，忽略了数据属于公司，出于各种原因数据管理者需要能够被灵活撤换的需求。例如：原数据管理者离职时需要向新的继任者移交数据管理权，而继任者需要对被移交的云存储数据进行完整性验证以避免责任纠纷。此外，考虑到在一个云存储系统中经过一段时间后用户的密钥可能更新，数据管理者需要使用新的密钥对存储在云上的数据进行完整性验证。这些不可避免的现实问题要求云存储数据完整性验证方案必须考虑支持用户可撤销。

本章内容由四部分构成：首先，4.1 节将对支持用户可撤销的云存储数据完整性验证研究作出概述；其次，4.2 节提出了一个支持用户可撤销的动态数据完整性验证方案 EDRPA，并对其进行了安全性分析与性能分析；再次，4.3 节在 EDRPA 方案的基础上进一步提出了一种新的改进方案 PRRPA，同样对其进行了安全性分析与性能分析；最后，4.4 节对本章内容进行小结。

4.1 支持用户可撤销的云存储数据完整性验证性研究概述

4.1.1 相关工作与研究动机

问题的出现： 随着云模式的普及，越来越多的用户将数据存储在云上，这样可以从本地数据存储和维护带来的昂贵开销和繁重任务中解脱出来，并享受云计算带来的各种便利。但是这种数据外包的模式面临着众多的安全挑战^[4-6,18,101,110]。因此，一系列旨在确保远端存储数据完整性的验证方案被提出，例如 Juels 等^[23]提出了一种证据可恢复的 POR 审计方案。Ateniese 等^[24]提出了一种名为 PDP 的数据持有性证明完整性验证方案。Shacham-Waters 利用短签名构造了有效的 POR 公开完整性验证方案^[25]。需要指出的是，这些方案并没有考虑对用户数据进行隐私保护。而在实际应用中，用户的数据可能存在泄露的危险。这些潜在风险将极大地影响方案在云计算应用中的安全性。从保护数据隐私的角度出发，用户可以委托一个第三方审计者 TPA 来保证他们存储数据的安全，但同时他们又不希望 TPA

的审计过程中由于未经授权的信息泄漏对他们的数据安全造成新的威胁^[100]。2010年, Wang 等^[28]提出了一项保护隐私的云存储数据公开完整性验证方案, 方案依托同态认证技术和随机掩饰码技术实现了隐私保护和公开完整性验证功能, 并利用双线性对聚合签名技术完成了数据完整性验证的批处理操作。

问题的凸显: 自 Wang 等人与 Zhu 等人提出了一系列经典的具有保护隐私功能的云存储数据公开完整性验证方案^[26-28,36-37,39,41-44]后。人们很快注意到之前几乎所有的完整性验证方案都是固定用户在计算云存储数据的完整性验证标签。即这些完整性验证方案, 要求在整个数据管理周期使用云存储服务的都必须是同一个用户。这是因为, 云存储数据完整性验证方案中的完整性验证标签是由用户的私钥签名生成, 然后在公共审计验证过程利用公开信息进行验证。这样的云存储数据审计验证模式在真实情况下是不现实的。一方面在一个审计系统中经过一段时间用户的公钥可能更新; 另一方面, 用户可能只是一个公司的数据管理者, 他可能因为各种原因而离职, 例如因为高薪而跳槽。因此, 出于现实考虑, 一个云存储数据完整性验证方案应该支持有效的用户撤销。

Wang 首先引入了共享云存储完整性验证问题, 提出了一个基于群签名的用户可撤销的自我审计验证方案^[46], 以及一些基于动态广播重签名方案和双向代理重签名^[105]的共享云用户可撤销公开完整性验证方案^[47-48]。随后 Yuan 等^[30]使用了一个类似的群签名技术提出了一个公开完整性验证方案。由于都涉及到群签名^[20]和广播加密技术^[106], 这些方案的效率都难以满足实际需求。

问题的现状: 2015 年 Wang 等^[49]提出了一个高效的用户可撤销公开完整性验证方案 (Panda 方案)。此方案借助代理重签名技术, 将不同用户的数据块签名转换为当前用户签名形式, 从而很好地满足了用户动态可撤销系统的云存储数据完整性验证需求, 这是此类问题当前的最优解决方案。但是 Wang 在文献[49]的方案局限性描述中, 提到云服务器与已撤销用户合谋可能会造成用户私钥的泄露。并在文献[49]中明确提出在下一步工作中希望通过一个多层次的代理重签名方案^[50]来解决这个问题。

4.1.2 方案的系统模型

为确保用户可撤销系统云存储数据的完整性验证始终由合法用户 (即当前用户) 的公钥来完成。接下来我们将首先分析基础云存储数据完整性验证模型, 接着进一步分析多用户云存储数据完整性验证模型, 最后构建出有效支持用户可撤销的具有隐私保护功能的云存储数据公开完整性验证模型。

4.1.2.1 基础云存储系统云数据存储模型

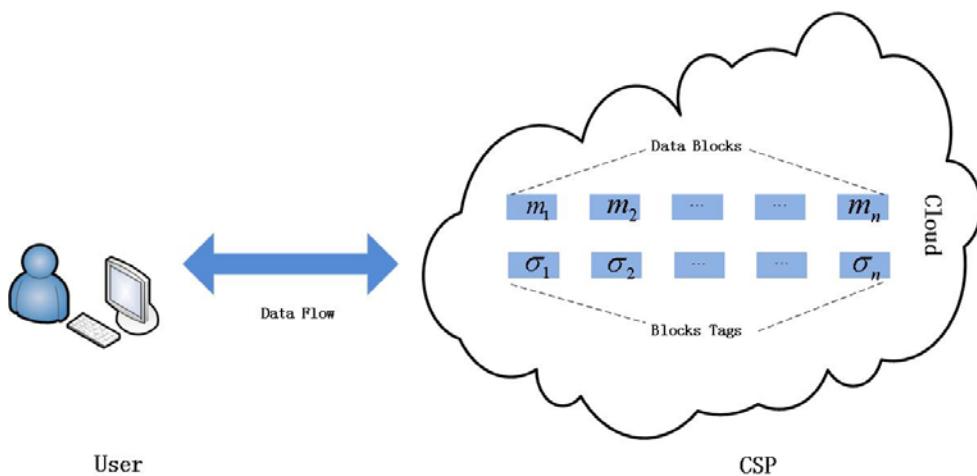


图 4-1 基础云存储系统数据存储模型

如图 4-1 所示，一个基本的云存储完整性验证系统包括两个主要实体：用户和云存储服务提供商（CSP）。用户可能是一个公司或组织（通常情况下，用户常常指公司和组织中将大量数据存储在云上并进行管理的数据管理员）。CSP 是云存储服务提供商，CSP 往往具有充沛的存储空间，并向用户提供经济和专业的存储服务。一个基础的云存储审计系统的工作过程具体描述如下：用户首先将数据 M ， $M \in \{0, 1\}^*$ 分成 n 块，这样每一个数据块可以表示成 $m_i \in Z_p$ ，其中 $i \in \{1, \dots, n\}$ 。同时用户利用自身的私钥对所有的数据 $M = (m_1, \dots, m_n) \in (Z_p)^n$ ，进行签名生成 $\sigma = (\sigma_1, \dots, \sigma_n)$ ，这里的签名即是数据的完整性验证标签。然后用户将所有的数据和相应的标签发送给云服务器，同时将本地相关数据删除。当需要验证外存数据的完整性时，用户挑选一系列随机的数据块并发送一个相应的 $Q = \{(i, v_i)\}$ 给云服务器，其中 i, v_i 表示数据块的 id 与随机序号。一旦收到 Q ，云服务器利用这些数据块和相应的标签计算并返回一个证据。最后，用户验证证据的有效性。如果验证失败，用户就可以确认云端数据已经损坏了；如果验证通过，则证明数据是完整的。用户可以重复“挑战-验证”这一过程直到最终确认存储在云端数据的完整性。从图 4-1 可以很明显的看到基础云存储系统云数据存储模型中云服务器只存储着单个用户的数据块和相应的数据验证标签。

4.1.2.2 支持用户可撤销的云存储系统多用户数据存储模型

如图 4-2 所示支持用户可撤销的云存储系统与基础的云存储系统有很大不同，其完整性验证方案主要实体涉及到多个用户。从现实考虑，数据属于公司，而非数据管理者。在某一个时期通常只有一个数据管理者正在对存储数据进行管理，但是某一段较长的时期内存储数据却可能被多个数据管理者先后进行过管理。即

一段时间后当前数据管理者可能不再适合管理存储在云上的数据，他可能被另一个管理者所替换。

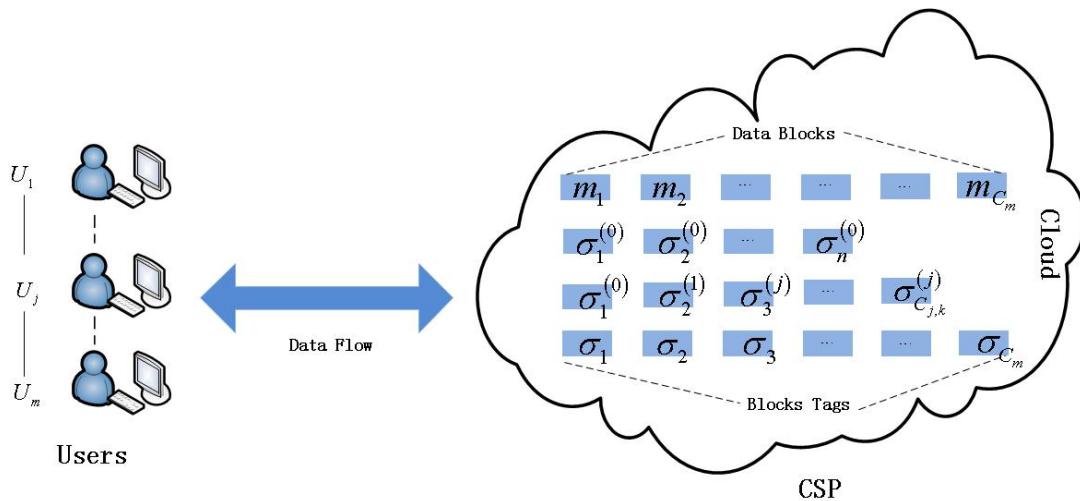


图 4-2 支持用户可撤销的云存储系统多用户数据存储模型

假设最开始有一个用户代表公司或组织将数据上传到云服务器，这个初始用户可以记做 U_0 ，然后公司雇佣数据管理者对存储的数据进行管理，显然地，从实际出发，数据管理者不可能是终身制的。在一个数据管理者离职前他需要向新的数据管理者移交数据，继任者需要对移交的数据进行审计，以便确认他的前任是否尽职的完成了数据管理工作。通常将数据外存在云上的数据存储模式是简单而有效的，公司或组织每一个时期只需要一个数据管理者就能很好的完成数据管理任务。按照时间先后将除 U_0 以外的数据管理者排序为 U_1, U_2, \dots, U_m ， m 为正整数。相应的，外存在云上的数据按照不同管理者的任期，数据管理周期被划分为 m 个周期 T_1, T_2, \dots, T_m 。在这里，显然每一个管理者只有在周期结束时才会向继任者移交数据。（注：对于第一个用户周期来说，用户 U_1 可能就是初始用户 U_0 ，但是 U_0 也可以委托一个新的用户 U_1 来完成数据管理工作，在文中我们统一分开表述。）

初始用户 U_0 首先将整个数据分成 n 个数据块，并用自己的私钥计算相应的完整性验证标签 σ ，然后他将整个数据和验证标签都上传到云服务器以完成数据的最初上传。 U_1 在 T_1 期间内进行数据管理，并在 T_1 周期结束时被 U_2 所取代， U_2 也将被 U_3 取代，一直到 U_j 取代 U_{j-1} ， $j \in \{0, \dots, m\}$ ， U_j 为当前数据管理者。因为完整性验证标签和用户相关，我们知道如果用户撤销了，标签也应当做出相应的修改。一个直接的做法就是先取回存储在云服务器上的数据块，再使用当前用户的私钥重新计算这些数据块的标签。然而，这并不是一个有效的支持用户可撤销的云存储完整性验证方案，因为这样将带来沉重的通信和计算负担。按照现实情况，每一个数据管理者都有可能在他的管理周期对公司和组织的数据进行添加、删除、修改等

数据动态操作，而且也只能在其相应的管理周期进行这些操作。例如对于当前用户 U_j 来说这些操作只能在周期 T_j 时完成。在添加操作中， U_j 将数据分成若干数据块，并用自己的私钥计算相应的存储标签并上传云存储服务器。如果是修改操作， U_j 先取回要修改的数据和相应的标签，经验证无误后，丢弃原来的标签，再修改数据块；并用自己的私钥计算新的完整性验证标签，并最后上传至云存储服务器。为简便起见，我们认为云存储服务器能够很好的处理当前数据管理者在当前管理周期进行的数据删除工作（比如说已删除数据在云服务器中全部记为“0”，若在完整性验证挑战中被抽中，并不会影响后面完整性验证的计算结果。但就实际而言这些已删除数据是不再占据存储空间的，不会带来额外的存储开销。）因此，不论何种操作云服务器上数据块的编号只会增加不会减少。

设对应每一个周期 T_1, T_2, \dots, T_m 结束，云存储服务器上数据块的增加总数量为 c_1, \dots, c_m ，其中 c_j 为正整数 $j \in \{0, \dots, m\}$ 。若记 C_1, \dots, C_m 为周期末时云服务器上数据块个数的总数，对于当前周期 T_j ，则云服务器上数据块总数为： $C_j = n + \sum_{\ell \in [1, j]} c_\ell = C_{j-1} + c_j$ ，其中 $\ell \in \{1, \dots, j\}$ 。若设当前周期用户 U_j 在周期内进行的所有添加操作所增加的数据块的个数为 $P_{j,1}, P_{j,2}, \dots, P_{j,\theta}$ ，则周期内增加的数据块总数为： $\sum_{k \in [1, \theta]} p_{j,k} = c_j$ ，其中 $p_{j,k}$ 为正整数， $k \in \{1, \dots, \theta\}$ 。若审计前用户 U_j 进行的最后一次添加操作为 $p_{j,k}$ ，则云服务器上需要审计的当前数据块总数为 $C_{j,k} = C_{j-1} + P$ ，这里为了方便表示我们记 $P = P_{j,1} + P_{j,2} + \dots + P_{j,k}$ ， $C'_j = C_{j,k}$ 文中将不对周期内的操作做过多的描述，审计当前云服务器上数据块的个数用 C'_j 表示。

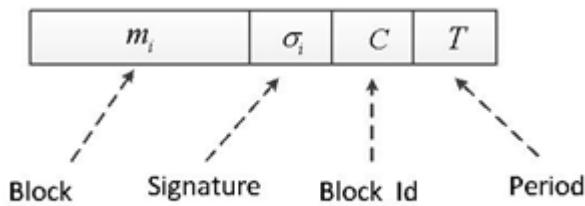


图 4-3 每一个数据块所对应的标签、数据块序号和管理周期

最终对于一个较为实用的用户可撤销云存储系统来说，存储在云上的数据将包括所有用户（初始用户 U_0 和此外的所有数据管理者 U_1, U_2, \dots, U_m ）上传的数据 m_1, \dots, m_{C_m} 以及相对应的数据标签 $\sigma_1, \dots, \sigma_{C_m}$ 和相应的周期 T_1, T_2, \dots, T_m 。如图 4-3 所示数据 m_i 的完整性将由 σ, c, T 来验证。

4.1.2.3 支持用户可撤销的隐私保护的公开云存储完整性验证模型

为了节省通信资源，比如说由于周期性数据完整性验证带来的潜在在线负担。云用户可以委托第三方审计者（TPA）执行安全审计案任务，因为这种模式是经济并可以自行处理的。与此同时，面向 TPA 与云服务器，云用户还可以保持数据的私密性。因此，第三方公开完整性验证方案更能满足真实环境的需求。如图 4-4 所示，支持用户可撤销的具有隐私保护功能的云存储数据第三方公开完整性验证方案“挑战-验证”过程如下：当用户想检验他的外存数据，他发送一个验证请求给 TPA。当 TPA 收到验证请求，它挑选一系列随机的数据块并发送一个相应的 $Q = \{(i, v_i)\}$ 给云服务器，其中 i, v_i 表示数据块的 id 与随机序号。一旦收到 Q ，云服务器利用这些数据块和相应的标签计算并返回一个证据。最后，TPA 验证证据的有效性。如果证据是无效的，TPA 就可以确认云端数据已经被损坏了；否则，数据仍然是完整的。用户可以重复“挑战-验证”过程直到最终确认云端数据的完整性。最后，TPA 将完整性验证方案结果返回用户。从图 4-4 中可以清楚的看到云服务器存储着所有用户的数据块和相应的数据标签。

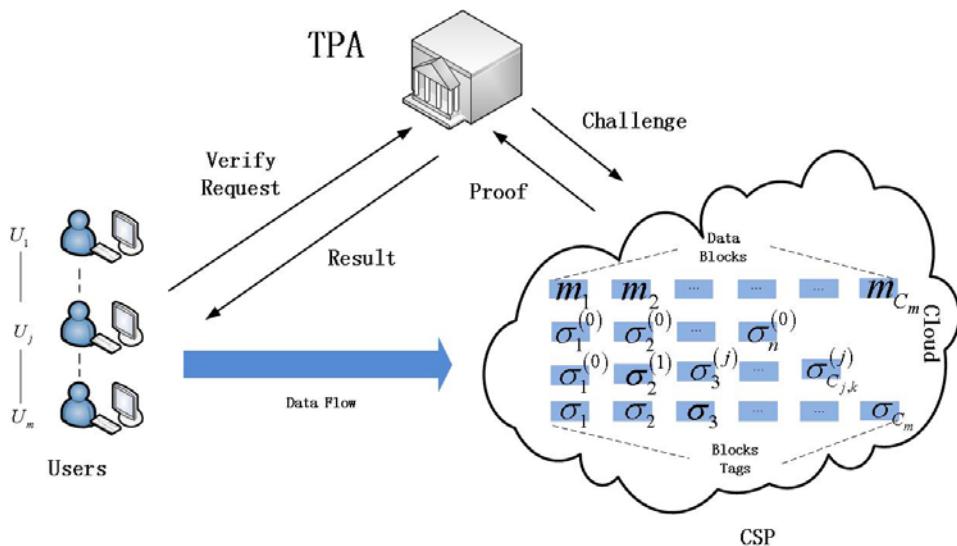


图 4-4 支持用户可撤销的具有隐私保护功能的云存储数据公开完整性验证模型

4.1.2.4 Panda 方案的问题

下面描述 Panda 中代理重签名方案实现签名转移的主体部分：

1. 在签名算法 $Sign$ 中，给定私钥 $sk_A = a$ ，数据块 $m \in Z_p$ 身份 id ，用户 u_A 输出基于数据块 m 的标签： $\sigma = (H(id) \omega^m)^a \in G_1$ 。
2. 在修改密钥生成算法 $ReKey$ 中，代理通过以下步骤生成一个重签名密钥 $rk_{A \rightarrow B}$ ：
 - (1) 代理生成一个随机的 $r \in Z_p^*$ ，并将他发送给用户 u_A ；

- (2) 用户 u_A 计算并发 r/a 给用户 u_B , 其中 $sk_A = a$;
 - (3) 用户 u_B 计算并发 rb/a 给代理, 其中 $sk_B = b$;
 - (4) 代理恢复出重签名密钥 $rk_{A \rightarrow B} = b/a \in Z_p^*$ 。
3. 在重签名算法中, 代理收到重签名密钥 $rk_{A \rightarrow B}$, 后执行代理重名:
 $\sigma' = \sigma^{rk_{A \rightarrow B}} = (H(id)\omega^m)^{a \cdot b/a} = (H(id)\omega^m)^b$ 。
4. 在验证算法 $Verify$ 中, 用户 u_B 通过验证公式 $e(\sigma', g) = e(H(id)\omega^m, pk_B)$ 对数据的完整性进行验证。

从上述过程中, 清晰可见重签名密钥 $rk_{A \rightarrow B}$ 的生成, 是基于用户 u_A 与 u_B 之间的私钥传递。虽然在这个过程中代理生成了一个密钥参数 $r \in Z_p^*$ 来对传递过程中的私钥进行保护。但是如果恶意云服务器(代理)与已撤销用户 u_A 合谋可以轻易计算出当前审计用户 u_B 的私钥 b , 因此方案存在用户私钥泄露的风险。

4.2 EDRPA 方案

EDRPA (Efficient dynamic integrity verification for big data supporting users revocability) 方案是我们基于 4.1 节问题分析所提出的一种有效的支持用户可撤销和具有隐私保护功能的支持第三方公开审计的云存储数据完整性验证方案。接下来我们将首先对其进行形式化定义, 构建安全模型。然后利用双线性对聚合签名技术提出一个有效和安全的公开审计方案。方案有效的实现了当前用户对在云上存储的所有历史用户的数据进行审计, 满足了由于用户离职而产生的数据移交需求。同时云用户可以委托第三方审计者执行安全审计任务。然后证明方案的安全性, 可以抵抗由已撤销用户和云服务商合谋而产生的攻击。最后, 实验结果和效率分析表明方案是高效的。

4.2.1 方案的形式化定义与安全模型

不失一般性的假设支持用户可撤销和具有隐私保护功能的云存储数据公开完整性验证方案包括个 $m + 1$ 认证用户 ($m \in Z > 0$), 其序列为 $U_0, U_1, U_2, \dots, U_m$, 相应的数据管理周期为 T_1, T_2, \dots, T_m 。(需要指出的是, 由于非认证用户可以被轻易识别, 而且他们也不可能损坏到外存的数据, 因此我们总是假设在我们的完整性验证方案中没有未经批准的用户。) 完整性验证方案可以定义如下:

定义 4.1 一个支持用户可撤销和具有隐私保护功能的云存储数据公开完整性验证方案由六个多项式算法 (PPT) 组成: ($Setup$, $SigGen$, $Update$, $Challeng$, $ProofGen$, $ProofVerify$)

1. 系统参数设置 ($Setup$): 该算法用于生成所有用户的公私钥对, 因此它可

由方案中任意用户 U_j 执行, $j \in \{0, \dots, m\}$ 。对于当前序列为 $j - th$ 的用户, 算法输入一个秘密参数 λ , 输出 U_j 的公私钥对 (pk_j, sk_j) 。

2. 标签生成算法 ($SigGen$): 该算法用于生成存储数据的标签。它由三个子多项式算法组成: $(SigGen(U_0), SigGen(U_j), SigGen(U_\ell \rightarrow U_j))$

初始标签生成算法 ($SigGen(U_0)$): 该算法用于在初始时刻生成初始的数据块标签, 因此它由方案中的初始用户 U_0 执行。算法输入 U_0 的私钥 sk_0 和数据块 (m_1, \dots, m_n) , 其中 $m_i \in \{0, 1\}^*$, $i \in \{1, \dots, n\}$; 并输出 (m_1, \dots, m_n) 与用户 U_0 相关的验证元 V 。然后 U_0 将 V 和 (m_1, \dots, m_n) 一起发送给云服务器, 最后从本地将它们删除。

当前用户标签生成算法 ($SigGen(U_j)$): 该算法用于生成在当前周期 T_j 中由当前操作 $p_{j,k'}$ (其中 $k \in \{1, \dots, \theta\}$, $k' \in \{1, \dots, k\}$) 所生成数据块的标签, 因此它由当前用户 U_j 执行。算法输入 U_j 的私钥 sk_j 和数据块 $(m_{C_{j,(k'-1)}+1}, \dots, m_{C_{j,k'}})$, 其中 $m_i \in \{0, 1\}^*$, $C_{j,k'}$ 为正整数; 并输出 $(m_{C_{j,(k'-1)}+1}, \dots, m_{C_{j,k'}})$ 与用户 U_j 相关的验证元 V 。然后 U_j 将 V 和 $(m_{C_{j,(k'-1)}+1}, \dots, m_{C_{j,k'}})$ 一起发送给云服务器, 最后从本地将它们删除。

当前用户修改标签生成算法 ($SigGen(U_\ell \rightarrow U_j)$): 该算法用于生成当前周期 T_j 中由用户 U_j 对先前任意用户 U_ℓ (其中 $\ell \in \{1, \dots, j\}$) 上传数据进行修改后新数据的数据完整性验证标签, 因此该算法由当前用户 U_j 执行。算法首先取回云服务器上属于用户 U_ℓ 的数据块 m_i 以及其相应的标签, 然后验证其是否有效。如果有效, 用户 U_j 使用新数据 m_i^* 代替 m_i (为便于表示, 在文中 m_i^* 仍然记录为 m_i)。随后, 算法输入 U_j 的私钥 sk_j 和数据块 m_i , 并输出 m_i 与用户 U_j 相关的验证元 V 。然后 U_j 将 V 和 m_i 一起发送给云服务器, 最后从本地将它们删除。

3. 密钥修改算法 ($Update$): 该算法用于用户更新, 它在旧周期末, 新周期开始前由新用户执行。假设用户 U_{j+1} 将替换用户 U_j , 用户 U_{j+1} 启用算法。当算法结束时用户将得到一个云服务器存储数据的修改密钥 $uk_{j \rightarrow j+1}$ 。最后用户将其上传到云服务器。

4. 挑战生成算法 ($Challeng$): 该算法用于用户生成挑战命令。假设当前用户 U_j 准备检验他的外存数据, 他首先发送一个验证请求到 TPA。当 TPA 收到验证请求, 它挑选一系列随机的数据块并发送一个相应的 $Q = \{(i, v_i)\}$ 给云服务器, 其中 i , v_i 表示数据块的 id 与随机序号。

5. 证据生成算法 ($ProofGen$): 该算法生成云服务器生成验证响应。算法输入数据块 $\{m_i\}_{i \in Q}$, $Challeng$ 和相应的验证元 V ; 输出一个验证证据 $proof$ 。

6. 证据验证算法 ($ProofVerify$): 该算法由 TPA 执行来验证证据 $proof$ 的正

确性。算法输入当前用户 U_j 的公钥 pk_j , $Challeng$ 和相应的 $proof$; 如果 $proof$ 有效输出VALID, 否则输出INVALID。最终TPA将审计结果返回给用户 U_j 。

定义 4.2 安全模型:

很容易理解, 支持用户可撤销的具有隐私保护功能的云存储数据公开完整性验证方案确保云端数据安全的直观定义如下: 如果云服务器存储数据确实损坏了, 而云服务器不承认, 甚至与已撤销用户合谋, 意图欺骗当前用户相信数据仍然是完整的。可假定云服务器为敌手 A 。为形式化云服务器与已撤销用户的合谋, 我们允许询问预言机, 输入一个已撤销用户的身份可以输出这个用户的私钥。但是我们禁止询问初始用户的身份, 因为云服务器与初始用户合谋在现实中是没有意义的。另外与通常的安全模型一致, 我们的安全模型同样允许 A 询问 $SigGen$ 预言机, $Update$ 预言机和 $ProofGen$ 预言机来获得相应的用户数据块标签, 修改密钥、有效证据以及挑战。

假如不存在恶意的云服务器(即概率多项式时间敌手 A)以不可忽略的优势在下述挑战者 B 与敌手 A 的交互游戏中获胜, 则称一个支持用户可撤销和具有隐私保护功能的云存储数据公开完整性验证方案是安全的。

1. 系统建立 ($Setup$): 挑战者通知敌手有关安全参数。挑战者 B 生成所有用户的公钥: $\{pk_j\}_0^m$, $j \in \{0, \dots, m\}$, 并将其发送给敌手 A 。

2. 询问 ($Query$): 敌手可以询问下列自适应模型。

签名询问 ($SigGen - Oracle$): 对于任意数据块 $m_i \in \{0, 1\}^*$, 如果敌手 A 想得到数据块的标签, 他可以用 m_i 询问签名预言机。一旦收到询问, 挑战者 B 执行算法 $SigGen(sk_j, m)$ 生成结果 V_j , 并返回 V_j 作为响应。

3. 撤销询问 ($Update - Oracle$): 当敌手 A 确认某用户不再适合执行完整性验证方案, 想用继任者 U_{j+1} 来代替先前用户 U_j , $j \in \{1, \dots, m-1\}$, A 询问预言机。一旦收到询问, 挑战者 B 利用 $Update(U_{j+1})$ 生成修改密钥 $uk_{j \rightarrow j+1}$, 并将它发送给 A 。 A 一旦收到 $uk_{j \rightarrow j+1}$, 可以生成数据块 m_i 的验证元 V_{j+1} 。 A 使用修改密钥 $uk_{j \rightarrow j+1}$ 将数据块 m_i 的验证元 V_{j+1} 与用户 U_{j+1} 关联起来。

4. 修改询问 ($Corrupt - Oracle$): 假设当前所有的已撤销用户为 U_0, U_1, \dots, U_d , $d \in \{0, \dots, m-1\}$ 。敌手 A 可以向预言机询问除了初始用户 U_0 以外的任意用户。当询问任意用户 U_ℓ , $\ell \in \{1, \dots, d\}$ 时, 挑战者 B 返回用户 U_ℓ 的私钥 sk_ℓ 作为响应。

5. 证据 ($Proof$): 为了验证云服务器中存储的数据是否完整。挑战者 B 生成一个随机挑战 $Chal$, 并询问敌手 A 以返回一个数据块 $\{m_i\}_{i \in Q}$ 与当前用户相关的证据 V 。当输入挑战 $Chal$, 数据块 $\{m_i\}_{i \in Q}$, 和相应的数据验证元 V 时, 敌手 A 输出一

个证据 $proof$ 作为响应。

6. 伪造 (*Forgery*): 当以上过程结束, 敌手 A 应对挑战 $Chal$ 输出的 $proof$ 满足以下条件时, 我们称敌手 A 赢得了游戏。

$$Verification(pk_\ell, chal, proof) \rightarrow Valid$$

数据块 m_i 不是原数据块。

4.2.2 方案的设计目标

为确保提出的方案在前述模式下是安全的, 方案必须达到下面的安全和效率目标。

1. 公开审计: 在不取回整个数据或是增加用户在线负担的前提下, 允许 TPA 验证云存储数据的正确性。
2. 正确存储: 确保不存在不诚信的云服务器在存储数块受损的情况下通过 TPA 的审计验证。
3. 隐私保护: 确保 TPA 不能够在完整性验证过程中以任何途径获取用户数据内容信息。
4. 用户可撤销: 如果用户撤销, 那么他的继任者可以有效的建立新的审计程序。
5. 合谋抵抗: 如果云存储数据被改变, 即使云服务器与已撤销用户合谋, 审计方案也应当以很高的概率查觉。
6. 有效性: 方案的计算、通信与存储代价应当尽可能的小。

4.2.3 EDRPA 方案

由于方案中涉及到较多参数, 为方便阅读, 现将主要参数陈列如下表 4-1。

EDRPA 方案如图 4-4 中所示, 存在一个半可信的 TPA, 半可信的 TPA 定义如下: 它会诚实的审计数据块的完整性, 但它同时是好奇的, 它会尝试在验证过程中获取用户的私密信息。方案由六个多项式算法 (PPT) 组成: $Setup$, $SigGen$, $Update$, $Challeng$, $ProofGen$, $ProofVerify$ 。

设 G 与 G_T 是两个阶为素数 p 循环群, g 是 G 的生成元。定义双线性映射 $e : G \times G \rightarrow G_T$; $H : \{0, 1\}^* \rightarrow G$, $h : \{0, 1\}^* \rightarrow G$, $f_{k_3} : \{0, 1\}^* \rightarrow Z_p$ 表示单向哈希函数。完整性验证方案具体执行如下:

1. $Setup$: 输入一个安全参数 λ , 对于每一个用户 U_j , $j \in \{0, \dots, m\}$ 执行下列步骤:

(1) 随机选择一个 $x_j \in Z_p$ 。

表 4-1 符号说明表

符号	表达意义
n	初始数据块的个数;
T_1, T_2, \dots, T_m	数据管理周期
T_j	用户 U_j 相应的当前管理周期;
C	审计当时所有数据块个数的总和: $C_{j,k} = C_{j-1} + p$ 其中 $p = p_{j,1} + p_{j,2} + \dots + p_{j,k}$
C_1, \dots, C_m	在各周期 T_1, T_2, \dots, T_m 末数据块序号 $i - th$ 的值
c_1, \dots, c_m	在各周期 T_1, T_2, \dots, T_m 末数据块的增量
$p_{j,1}, p_{j,2}, \dots, p_{j,\theta}$	在周期 T_j 中各操作 $P_{j,1}, P_{j,2}, \dots, P_{j,\theta}$ 引起的数据块增量
$\sigma_i^{(j)}$	表示对应于数据块 m_i 由用户 U_j 生成的标签
Q	$Q = \{(i, v_i)\}$ 其中 i 是数据块的序号, 而 v_i 是随机数。 Q 是随机选取的一系列 i 和 v_i 的集合。用来指定将要挑战的数据块和数据块序号
t	用来验证数据块的序号 $i - th$ 是否与数据块相匹配
V	挑战 Q 的响应

(2) 计算公钥 g^{x_j} 。

(3) 输出 $pk_j = g^{x_j}$ 和 $sk_j = x_j$ 。

2. $SigGen$: 该算法用于生成存储数据的标签。它由三个子多项式算法组成:

$(SigGen(U_0), SigGen(U_j), SigGen(U_\ell \rightarrow U_j))$

$SigGen(U_0)$: 公司或组织在最初的时间周期委派一个可信的用户 U_0 将初始数据上传云服务器。

(1) 初始用户 U_0 重编码所有的文件并将它们分成 n 个数据块, 其中任意数据 $m_i \in Z_p$, $(m_1, \dots, m_n) \in (Z_p)^n$ 。

(2) 对于所有的数据块 $\{m_i\}_{1 \leq i \leq n}$, 用户计算每个数据块 m_i 相应的数据标签 $\sigma_i = (H(W_i) u^{m_i})^{x_0}$, 对数据块序号 $i - th$ 计算相应的验证标识 $t_i = W_i || Sig_{ssk}(W_i)$, 这里 u 是一个从 G 中公开选择的随机参数, $W_i = i || T_j$, $j \in \{1, \dots, m\}$ 。

(3) 发送初始验证元 $V = \{\sigma_i, t_i\}_{1 \leq i \leq n}$ 和数据块 $\{m_i\}_{1 \leq i \leq n}$ 到云服务器, 然后在本地将其删除。

$SigGen(U_j)$: 该算法用于生成在当前周期 T_j 中由当前操作 $p_{j,k'}$ (其中 $k \in \{1, \dots, \theta\}$, $k' \in \{1, \dots, k\}$) 所生成数据块的标签。

(1) 当前用户将当前操作新增数据处理为 $\{m_i\}_{C_{j,(k'-1)} \leq i \leq C_{j,k'}}$ 。这里 $C_{j,k'}$ 是一个正整数, 表示在周期 T_j 用户 U_j 由于操作 $p_{j,k'}$ 向云服务器外存数据块个数增加的增量。

(2) 对于所有的数据块 $\{m_i\}_{C_{j,(k'-1)}+1 \leq i \leq C_{j,k'}}$, 用户计算每个数据块 m_i 相应的数据标签 $\sigma_i = (H(W_i) u^{m_i})^{x_j}$, 对数据块序号 $i - th$ 计算相应的验证标识 $t_i = W_i || Sig_{ssk}(W_i)$, 这里 u 是一个从 G 中公开选择的随机参数, $W_i = i || T_j$, $j \in \{1, \dots, m\}$ 。

(3) 发送初始验证元 $V = \{\sigma_i, t_i\}_{C_{j,(k'-1)}+1 \leq i \leq C_{j,k'}}$ 和数据块 $\{m_i\}_{C_{j,(k'-1)}+1 \leq i \leq C_{j,k'}}$ 到云服务器, 然后在本地将其删除。

$SigGen(U_\ell \rightarrow U_j)$: 该算法用于生成当前周期 T_j 中由用户 U_j 对先前任意用户 U_ℓ (其中 $\ell \in \{1, \dots, j\}$) 上传数据进行修改后新数据的数据块标签。

(1) 如果当前用户 U_j 因为某些原因想修改先前周期 T_ℓ 中的数据块, 则他必须先回检 m_i 和 $V = \{\sigma_i, t_i, C_\ell\}$ 。

(2) 随后用户 U_j 用先前用户的公钥验证 $t_i = W_i || Sig_{ssk}(W_i)$ 是否有效。如果无效则完整性验证方案终止, 如果有效 U_j 用新数据 m_i^* 代替 m_i 并仍然记录为 m_i , 与此同时将数据标签 $\sigma_i^{(\ell)} = (H(W_i) u^{m_i})^{x_\ell}$ 替换为 $\sigma_i^{(j)} = (H(W_i) u^{m_i})^{x_j}$; 将序号 $i - th$ 验证标识 $t_i = (i || T_\ell) || Sig_{ssk(U_\ell)}(i || T_\ell)$ 替换为 $t_i = (i || T_j) || Sig_{ssk(U_j)}(i || T_j)$ 。

(3) 最后, 用户 U_j 发送验证元 $V = \{\sigma_i, t_i\}$ 和数据块 m_i 到云服务器, 然后在本地将其删除。

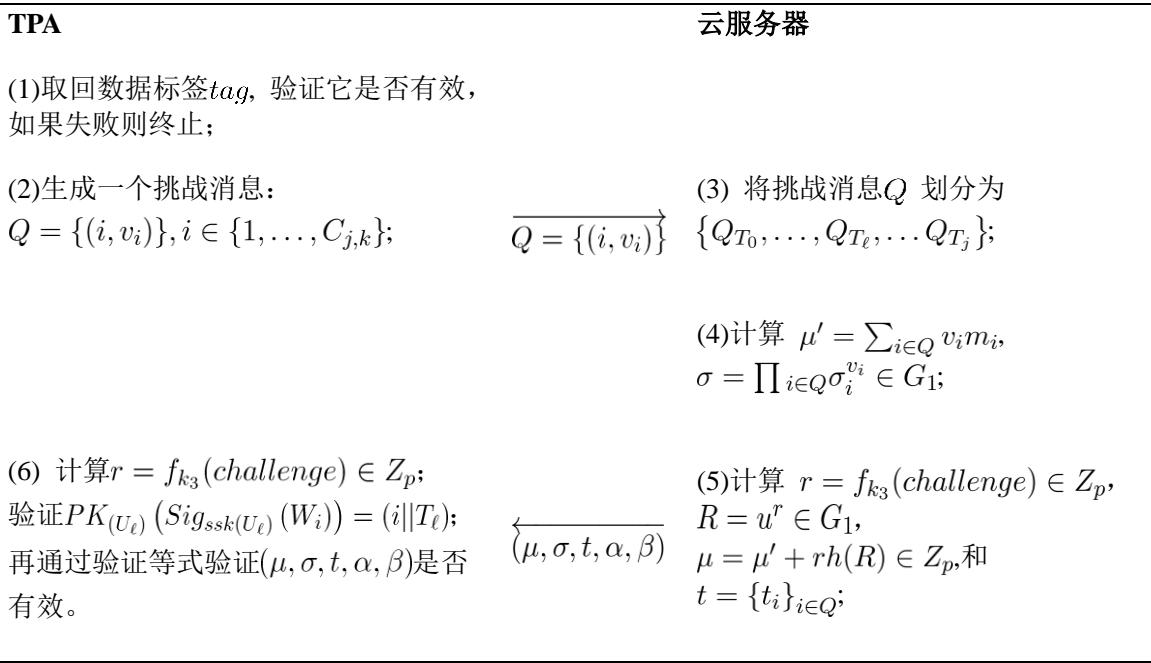


图 4-5 数据完整性验证过程

3. *Update*: 当用户 U_{j+1} 将取代用户 U_j 执行数据管理, 用户计算修改密钥 $uk_{j \rightarrow j+1}$, $uk_{j \rightarrow j+1} = (g^{x_j})^{\frac{1}{x_{j+1}}} = g^{x_j/x_{j+1}}$, 并将它发送给云服务器。云服务器执行

下列步骤:

(1) 设置 $\alpha_j = pk_j$, $\beta_{j+1} = uk_{j \rightarrow j+1}$ 。

(2) 对于任意的 $j \in \{0, \dots, m-1\}$, 用 $V_i = (\sigma_i, t_i, \alpha_1, \dots, \alpha_j, \beta_1, \dots, \beta_j)$ 将数据块 m_i 的验证元与用户 U_j 关联起来。

4. *Challeng*: 如果当前用户 U_j 在周期 T_j (U_j 的管理周期) 中准备检验他的外存数据, 他首先发送一个验证请求到 TPA。方案主体完整性验证过程如图 4-5 所示:

5. *ProofGen*: 当 TPA 收到用户的验证请求, 它将挑选一系列随机的数据块并发送一个相应的 $Q = \{(i, v_i)\}$ 和通信密钥 k_3 给云服务器作为 *Challeng*, 其中 i 是数据块的序号, 而 v_i 是随机数。一旦收到 *Challeng*, 云服务器将 Q 分为 $\{Q_{T_0}, \dots, Q_{T_\ell}, \dots, Q_{T_j}\}$, 然后计算并返回 $(\mu, \sigma, t, \alpha, \beta)$ 作为 *proof*。

这 里 有 $r = f_{k_3}(\text{challenge}) \in Z_p$, $R = u^r \in G$, $\mu' = \sum_{i \in Q} v_i m_i$, $\mu = \mu' + rh(R) \in Z_p$, $\sigma = \prod_{i \in Q} \sigma_i^{v_i} \in G$ 和 $t = \{t_i\}_{i \in Q}$ 。

6. *ProofVerify*: 当 TPA 收到 *proof*, 算法输入用户 U_ℓ 的公钥 pk_ℓ , 挑战 $Q = \{(i, v_i)\}$ 与 k_3 , 证据 $(\mu, \sigma, t, \alpha, \beta)$; 当下列等式同时成立时算法输出 *VALID*。并最终由 TPA 将完整性验证方案结果返回给用户 U_j 。

首先对于每一个 t_i 验证:

$$PK_{(U_\ell)}(Sig_{ssk(U_\ell)}(W_i)) = (i || T_\ell) \quad (4-1)$$

其次验证数据块标签:

$$e(\sigma, g) = \prod_{\ell \in [0, j]} e\left(\alpha_\ell, u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \quad (4-2)$$

最后验证用户:

$$e(\alpha_{j-1}, g) = e(pk_j, \beta_j) \quad (4-3)$$

$$e(\alpha_\ell, g) = e(\alpha_{\ell+1}, \beta_\ell) \text{ for } \ell \in \{0, \dots, j-2\} \quad (4-4)$$

具体验证执行过程如上述图 4-5:

从上述步骤中, 我们可以看出, EDRPA 方案的修改过程在计算和通信代价方面是简单有效的, 因为它只需要计算和发送一个修改密钥 $uk_{l \rightarrow l+1}$ 。此外, 由于 EDRPA 方案在当前周期时间只需要使用当前用户的公钥进行完整性验证。完整性验证过程并不需其它所有已撤销用户的公钥 $pk_j = g^{x_j}$ 参与, 因此需要对服务器发送给云服务器的 *proof* 中的 α_ℓ 在公式 (4-4) 中进行验证。这可以有效阻止云服务器与已撤销用户合谋对其进行任意修改。

4.2.4 安全性证明

4.2.4.1 正确性

定理 4.1: 方案满足正确性。

证明: 根据前面的内容, 可以看到对于任何挑战 $Q = \{(i, u_i)\}$, 服务器会先将其分割为 $Q = \{Q_{T_0}, \dots, Q_{T_\ell}, \dots, Q_{T_j}\}$, $\ell \in \{0, \dots, j\}$, 有:

$$\begin{aligned} e(\sigma, g) &= e\left(\prod_{i \in Q} \sigma_i^{v_i}, g\right) \\ &= \prod_{\ell \in [0, j]} e\left(\prod_{i \in Q_{T_\ell}} \sigma_i^{v_i}, g\right) \\ &= \prod_{\ell \in [0, j]} e\left(\prod_{i \in Q_{T_\ell}} (H(W_i) u^{m_i})^{v_i}, g^{x_\ell}\right) \\ &= \prod_{\ell \in [0, j]} e\left(\alpha_\ell, u^{\mu - rh(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \\ &= \prod_{\ell \in [0, j]} e\left(\alpha_\ell, u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \end{aligned} \quad (4-5)$$

同样对于任何用户 U_j , $j \in \{1, \dots, m\}$ 有:

$$e(\alpha_{j-1}, g) = e(g^{x_{j-1}}, g) = e\left(g^{x_j}, g^{\frac{x_{j-1}}{x_j}}\right) = e(pk_j, \beta_j) \quad (4-6)$$

对于所有的 $\ell \in \{0, \dots, j-2\}$, 有:

$$e(\alpha_\ell, g) = e(g^{x_\ell}, g) = e\left(g^{x_{\ell+1}}, g^{\frac{x_\ell}{x_{\ell+1}}}\right) = e(\alpha_{\ell+1}, \beta_\ell) \quad (4-7)$$

经上证明: 完整性验证方案是正确的。 ■

4.2.4.2 安全性分析

下面首先证明该方案能够抵抗恶意云服务器通过产生伪造的审计证明响应信息来欺骗 TPA 的完整性验证过程。

定理 4.2: 方案满足不可伪造性。恶意的云服务器产生伪造的审计证明响应信息 $Proof$ 来欺骗 TPA 的完整性验证过程在计算上是不可行的。

证明: 假设存在恶意的云服务器 (多项式时间敌手 A) 以不可忽略的概率 ξ 产生伪造的审计证明响应信息来欺骗 TPA 的完整性验证过程。下面设置多项式时间算法 (挑战者 B) 通过运行敌手 A 作为子程序也以不可忽略的概率 ξ 解决 CDH 困难性问题。算法 B 与敌手 A 的信息交互如下:

Setup: 给定一个安全参数 λ , 算法 B 首先随机选取 G 的一个生成元 g , 这里 $g^\alpha \in G$, $H : \{0, 1\}^* \rightarrow G$ 将在证据里被模式化为随机预言模型。对于未知的 x_0 , B 也可以随机地从 G 中选取 g^{x_0} 作为用户 U_0 的公钥, 同样对于 $j \in \{0, \dots, m\}$, 从 Z_q 中选取 x_j 计算用户 U_j 的公钥 g^{x_j} 。然后 B 设置 $u = g^\alpha$, 最后将系统参数 g 、 u 和所有的用户

公钥 $\{g^{x_j}\}_0^m$ 发送给敌手A。

Query. 敌手A可以询问以下自适应模型。假设，对任意数据块 m_i ，首先作一个H-Oracle查询。

H-Oracle: 当B向预言机询问数据块 m_i ，B在H-list中查询对应的元组 $(m_i, s_i, H(W_i))$ 。如果B找到了一个匹配对，输出响应 $H(W_i)$ ；否则B首先选择一个随机值 $s_i \in Z_p$ 然后计算 $H(W_i) = g^{s_i}/u^{m_i}$ ，将 $(m_i, s_i, H(W_i))$ 存储于H-list最后输出响应 $H(W_i)$ 。

SignGen-Oracle: A向预言机询问获取数据块 $\{m_i\}_{i \in [1, C_{j,k}]}$ 的标签。一旦收到询问，对于 $i \in \{1, \dots, C_{j,k}\}$, $j \in \{1, \dots, m\}$, $\ell \in \{0, \dots, j\}$, B在H-list中查询 m_i 寻找匹配元组并计算 $\sigma_i = (g^{x_\ell})^{s_i}$ 最终输出一系列 $V = (\sigma_1, \dots, \sigma_{C_{j,k}})$ 作为响应。因为 $\sigma_i = (H(W_i)u^{m_i})^{x_\ell}$ 代入等式得到 $H(W_i) = g^{s_i}/u^{m_i}$ 。可以得到对于 $i \in \{1, \dots, C_{j,k}\}$, $H(W_i) = g^{s_i}/u^{m_i}$ 。

Update-Oracle: 如果A想用继任者 U_{j+1} 来代替用户 U_j , $j \in \{1, \dots, m-1\}$, A将询问 U_j . 一旦收到询问，B首先利用 U_{j+1} 的私钥 x_{j+1} 计算修改密钥 $uk_{j \rightarrow j+1} = g^{x_j/x_{j+1}}$ 。并将结果反馈给A。然后A设置 $\alpha_j = g^{x_j}$ 和 $\beta_{j+1} = uk_{j \rightarrow j+1}$ 并将其添加到用户的验证元组。则用户 U_j 的新验证元组为 $V_j = (\mu, \sigma, t, \alpha_1, \dots, \alpha_{j-1}, \beta_1, \dots, \beta_j)$ ，同样用户 U_{j+1} 的验证元组为 $V_{j+1} = (\mu, \sigma, t, \alpha_1, \dots, \alpha_j, \beta_1, \dots, \beta_{j+1})$ 。

Corrupt-Oracle: 设置所有的已撤换用户为 U_1, \dots, U_d ，这里 $d \in \{1, \dots, m-1\}$ 。如果敌手A针对 U_j 询问预言机，这里 $j \in \{1, \dots, d\}$ ，B就返回 U_j 的私钥 x_j 作为响应。

Proof: 如果B想验证数据 m 是否仍然完整地存储在云上，他将生成一个随机挑战 $Challeng(Q, k_3)$ ，其中 $Q = \{(i, v_i)\}$ 给A，这里 $i \in \{1, \dots, C_{j,k}\}$, $v_i \in Z_p$ 。令 $(m_1, \dots, m_{C_{j,k}}) \in (Z_p)^n$ ，当前用户为 U_j 一旦收到 $Challeng(Q, k_3)$ ，A计算 $r = f_{k_3}(challenge) \in Z_p$, $R = u^r \in G$, $\mu' = \sum_{i \in Q} v_i m_i$, $\mu = \mu' + rh(R) \in Z_p$, $\sigma = \prod_{i \in Q} \sigma_i^{v_i}$, $t = \{t_i\}_{i \in Q}$ 并生成一个有效的证明给B，这个证明是 $V_j = (\mu, \sigma, t, \alpha_1, \dots, \alpha_{j-1}, \beta_1, \dots, \beta_j)$ 。

Forgery: A以不可忽略的概率 ξ 输出一个有的效证据 $(\mu^*, \sigma^*, t, \alpha_1, \dots, \alpha_{j-1}, \beta_1, \dots, \beta_j)$ ，作为用户 U_j , $j \in \{1, \dots, m\}$ 对已损坏数据块 $\{m_i\}_{i \in [1, C_{j,k}]}$ 的挑战 $Challeng(Q, k_3)$ 的响应。令数据块未损坏的情况下原本 U_j 挑战 $\{m_i\}_{i \in [1, C_{j,k}]}$ 响应 $Challeng(Q, k_3)$ 的输出证明为 (μ, σ) ，这里显然 $\mu \neq \mu^*$ 。令 $\Delta\mu = \mu^* - \mu$ 由于

$$e(\sigma, g) = \prod_{\ell \in [0, j]} e\left(\alpha_\ell, u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \quad (4-8)$$

$$e(\sigma^*, g) = \prod_{\ell \in [0, j]} e\left(\alpha_0, u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \quad (4-9)$$

从定义 4.2 我们可以得到 $e(\sigma^* \cdot \sigma^{-1}, g) = e(g^{x_0}, u^{\Delta\mu})$ 由此结果我们可以推导出 $u^{x_0} = g^{ax_0} = (\sigma^* \cdot \sigma^{-1})^{\frac{1}{\Delta\mu}}$ 。因此，给定 $g, g^a, g^{x_\ell} \in G$ ，输出 σ^* 将以不可忽略的概率解决 CDH 问题。■

下面我们证明方案满足隐私保护性：

定理 4.3: 给定一个来自云服务器的审计证明响应信息 $Proof$ ，对于好奇的 TPA，它试图从中恢复用户数据文件 $F = \{m_1, m_2, \dots, m_n\}$ 中的数据块是不可行的。

证明：组合信息 $\mu' = \sum_{i \in Q} v_i m_i$ 是关于用户原始数据块的线性组合，一旦这个组合信息发送给 TPA，这个好奇的 TPA 可以通过收集大量的组合信息，并借助强大的计算设备来求解这些线性方程组，从而恢复用户的原始数据块。为了防止 TPA 读取用户的原始数据块信息， $\mu' = \sum_{i \in Q} v_i m_i$ 需要使用随机掩饰码技术，具体如下：

云服务器利用伪随机函数 f 计算随机值 $r = f_{k_3}(challenge) \in Z_p$ ，并计算 $R = u^r \in G$ ，最后将 $\mu' = \sum_{i \in Q} v_i m_i$ 盲化为 $\mu = \mu' + rh(R) \in Z_p$ 这样为了让 TPA 仍能解这些线性方程，TPA 必须掌握这个随机值 r ，进而它必须掌握这个伪随机函数的秘密密钥。事实上，这个秘密密钥只有云服务器知道，因此 TPA 不可能知道这个随机值 r 。因此，对于好奇的 TPA，它试图从审计证明响应信息中恢复用户数据文件是不可行的。■

4.2.5 效率分析

在本节我们分析 EDRPA 方案的通信和计算复杂度。需要注意的是与其它公共完整性验证方案^[26,28,46-49]一样我们只计算频繁审计活动中通信和计算代价而不计算系统建立时的通信与计算代价。

我们用 $Pair$ 表示双线性对操作， Exp 表示 G 上的指数操作， MZ 和 MG 分别表示 Z_p 和 G 上的乘法操作。 $|C|$ 、 $|n|$ 、 $|p|$ 、 $|G|$ 分别表示 $\{1, \dots, C_{j,k}\}$ 、 $\{1, \dots, n\}$ 、 Z_p 和 G 的比特长度。挑战中选取的数据块假定是个常量 c ，挑战中已撤换的用户数量总和也假设是个常量 d 。

表 4-2 Panda 方案^[49]与 EDRPA 方案的效率比较

方案	通信开销	计算开销		验证时间	合谋抵抗	公开审计
		修改密钥				
Panda 方案 ^[49]	$d \cdot (p + G) + c \cdot (id + n + p)$	$nExp$	$(c + 2d) MZ + dMG + (c + d) Exp + (d + 1) Pair$		不	是
EDRPA 方案	$ p + G + c(2 C + p)$	Exp	$(c + 2d) MZ + jMG + (2c + d) Exp + (d + 1) Pair$		是	是

1. 通信开销: 可以看到 EDRPA 方案中通信负载主要取决于审计证明响应信息的通信代价。其中发送挑战 $Q = \{(i, v_i)\}$ 到云服务器的通信量为 $c(|C| + |p|)$ 。云服务器返回审计证明响应信息 $(\mu, \sigma, t, \alpha_1, \dots, \alpha_{j-1}, \beta_1, \dots, \beta_j)$ 的通信量为 $|p| + 2j|G| + c|C| + |G|$, 但是我们注意到只有在用户 U_j 发起的第一次审计请求中才需要这样的通信量, 否则只需要传送通信量为 $|p| + |G| + c|C|$ 的 (μ, σ, t) 就行了。因此, 最终在一次完整性验证过程中总共的通信开销为 $|p| + |G| + c(2|C| + |p|)$ 。

2. 计算开销: 计算开销包括修改密钥生成、代理重签名和验证开销三部分, 由于本方案支持第三方公开审计, 所以只需要考虑修改密钥生成和验证开销两部分。对于用户 U_j , 在 $Update$ 算法中修改密钥生成只需要计算一次 g^{x_{j-1}/x_j} , 因此计算代价为 Exp 。根据上面提到的计算原则, 并简化加操作和哈希操作后我们计算出验证开销为: $(c + 2j) MZ + jMG + (2c + j) Exp + (j + 1) Pair$ 。

3. 开销比较: 如表 4-2 所示将 2015 年 Wang 等人^[49]的 Panda 方案与本文中的 EDRPA 方案进行对比。Panda 方案虽然在合谋攻击下是不安全的, 但仍然是文献 [49] 中最有效的支持用户可撤销的云存储数据公开完整性验证方案。当用户 U_j 执行 [49] 中的算法 $ProofGen$, 用户 U_j 发送一个规模为 $c(|n| + |q|)$ 的挑战 $Q = \{(i, v_i)\}$ 给云服务器。云服务器返回一个规模为 $j \cdot (|p| + |G|) + c \cdot |id|$ 的证据 $\{\alpha, \beta, \{id_l, s_l\}_{l \in L}\}$ 给用户 U_j 。因此, 一个完整性验证过程的全部通信开销为 $j \cdot (|p| + |G|) + c \cdot (|id| + |n| + |p|)$ 。文献[49]Panda 方案中 $Update$ 算法需要重新计算所有 n 个数据块的数据标签, 因此其修改计算代价为 $nExp$ 。每完成一次完整性验证 Panda 方案都必须先请求云服务器输出一个证据 (μ, σ) , 然后交由完整性验证方案者验证是否有效。由此可知在同样简化加操作和哈希操作后, 在文献[49] 中 Panda 方案的任意一个用户完成一次完整性验证的时间代价为 $(c + 2j) MZ + jMG + (c + j) Exp + (j + 1) Pair$ 。如表 4-2 所示 EDRPA 方案的完整性验证时间近似等于 Panda 方案, 修改时间要远小于 Panda 方案, 因此整体计算开销要小于 Panda 方案。同样如表 4-2 所示 EDRPA 方案的通信代价亦小于 Panda 方案。因此 EDRPA 方案对比 Panda 方案在效率上更具有优势。

4.2.6 实验分析

由上述可知, 两个方案计算开销的比较结果是明显的。EDRPA 方案的修改时间为 Exp 要远小于 Panda 方案中的 $nExp$ 。在完整性验证时间方面 EDRPA 方案与 Panda 方案近似, 只相差一个 $cExp$, 所以我们在实验中只需要对完整性验证过程中的通信开销进行比较。

实验平台如下: 实验设备的操作系统为 windows 7, 硬件配置是 Intel Core 2 i5

处理器, CPU 的频率为 2.53 GHz, 内存为 2G DDR 3 of RAM(1.74 GB available)。本节使用的是 MIRACL 库, 版本号为 5.6.1, 开发环境为 Microsoft Visual C 6.0。实验仿真所使用的椭圆曲线为 MNT 曲线, 基本的文件大小为 159 比特, 嵌入阶为 6。选择 $|p| = |q| = 160$ 比特, 并简单化处理设置 $k = 20$, $c = 300$ 。如图 4-6 所示, 实验表明, 与 2015 年 Wang 等人在文献[49]中提出的 Panda 方案对比, 本文 EDRPA 方案的通信代价要更低一些。

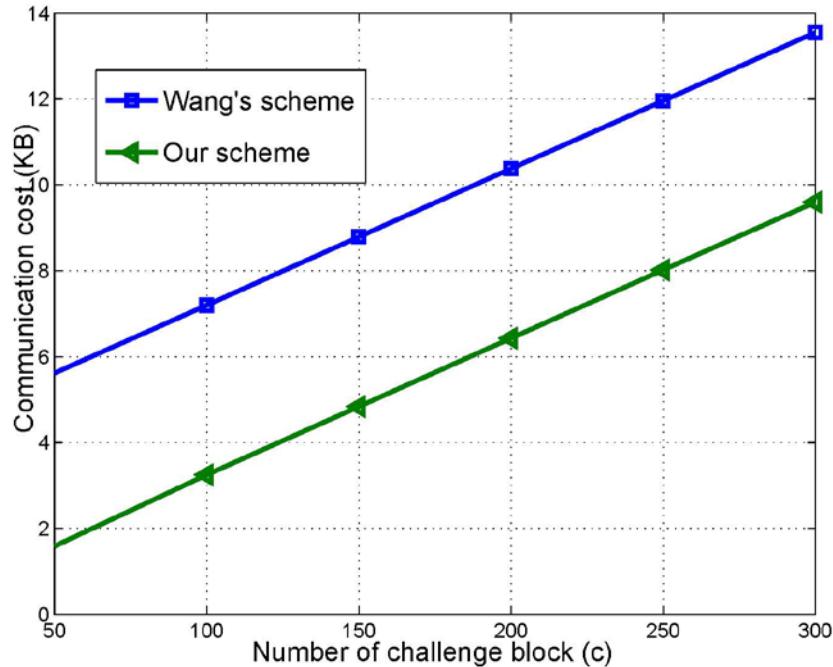


图 4-6 通信代价实验比较

4.3 改进方案 PRRPA

上述 EDRPA 方案虽然能够解决恶意云服务器与已撤销用户合谋的攻击问题, 但是我们发现它的效率并不尽如人意。比如 4.2 节中效率分析表明, 其验证时间代价要略高于 Panda 方案一个 $cExp$ 。另外, 其抗合谋攻击的功能具有一定的局限性, 方案虽然能够抵制已撤销用户与恶意云服务器合谋修改其他用户生成的数据, 但是却无法阻止已撤销用户与恶意云服务器合谋对自身签名数据进行修改。Wang 在文献[49]中明确提出在下一步工作中希望通过一个多层次的代理重签名方案来解决合谋问题。因此我们尝试从 Wang 的这个思路出发, 针对用户动态可撤销需要新的数据管理员对其前任所管理的数据进行完整性验证的需求, 基于单向代理重签名技术提出了一个具有隐私保护功能的支持用户可撤销的公共云存储数据完整性验证新方案 (Efficient public auditing scheme for cloud storage supporting user revocability with proxy re-signature scheme, PRRPA)。首先, 方案中所采用的单向代

理重签名算法，其代理重签名密钥由当前用户私钥结合已撤销用户公钥生成，不存在私钥泄露问题，这样能够安全的实现数据所有权的转移；其次，方案中，证明了恶意的云服务器不能产生伪造的审计证明响应信息来欺骗第三方审计者通过完整性验证过程；并且更进一步，方案采用了随机掩饰码技术能够有效防止好奇的第三方审计者恢复原始数据块。性能分析表明，与 2015 年 Wang 等人提出的 Panda 方案相比，此方案在增加新功能的基础上，其完整性验证过程中通信开销与计算代价仍全部低于 Panda 方案。

4.3.1 基本模型

如图 4-7 所示，PRRPA 方案涉及到多个用户。按照时间先后对除 U_0 以外的数据管理者排序为 U_1, U_2, \dots, U_m ， m 为正整数。相应数据管理周期被划分为 T_1, T_2, \dots, T_m 。显然每一个用户只有在周期结束时才会向继任者移交数据。初始用户上传数据以及初始数据块完整性验证标签的生成与 EDRPA 方案相同；但最终云服务器上不再存储所有用户的 data 标签，而是使用代理重签名技术，将所有存储在云服务器上的 data 标签转换成当前用户签名的模式。

最终对于当前用户来说，云服务器上存储的数据 的标签 最后都会转化为 $\sigma_i^{(j)}$ 。考虑到云用户构建一个进行数据无误性检验的环境是不可行和代价高昂的。因此为了节省进行存储数据周期无误性验证的通信资源和在线负担，云用户可以委托第三方审计者 (TPA) 来执行安全完整性验证任务，因为他是经济并且可以自动运行的。但是，云用户同时希望对 TPA 保持数据的隐私性。为了方便区分，本节用 $\sigma_i^{(j)}$ 表示用户 U_j 用自身的私钥对 m_i 签名生成的数据验证标签。

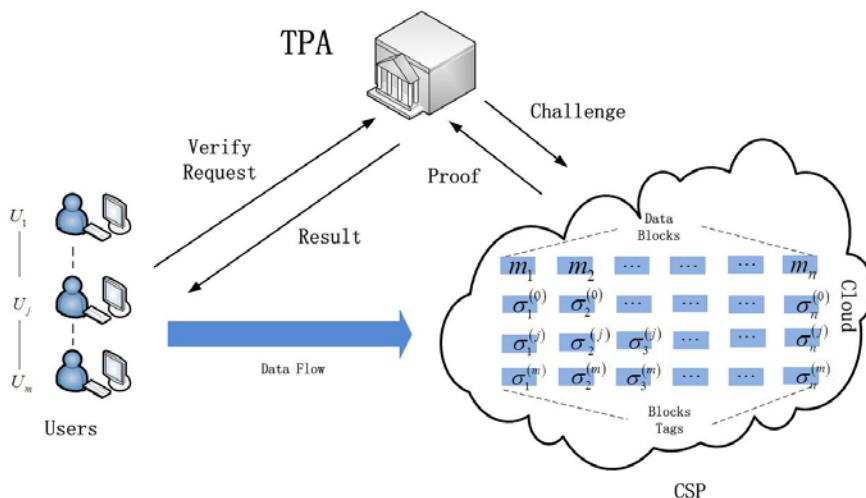


图 4-7 PRRPA 方案系统模型

4.3.2 代理重签名

为实现前面基本模型中所描述的支持用户动态可撤销的这一功能，并能持续地保证云服务器上存储数据的完整性，基于文献[50]中的代理重签名算法，做出了进一步的改进。通过当前用户私钥结合已撤销用户公钥生成新的代理重签名密钥来保证即使云服务器与已撤销用户合谋也无法影响数据的安全性。新的代理重签名算法和步骤与文献[50]相似，其中最重要的签名步骤如下：

Sign(x_i, m): 数据管理者 U_i 利用其私钥 x_i 对数据 $m \in \{0, 1\}^*$ 签名，生成 $\phi_i = H(m)^{x_i} \in G$ 。并将公钥 g^{x_i} 、生成元 g ，一起作为签名验证信息，记为：

$$\varphi_i = (\phi_i, \varepsilon_i, \gamma_i) = (H(m)^{x_i}, g^{x_i}, g) \quad (4-10)$$

ReKeygen: 当前数据管理者 U_j 利用其私钥 x_j 结合已撤销管理者公钥 g^{x_i} ，生成代理重签名修改密钥 uk_j ，并将其发送给云服务器：

$$uk_j = g^{x_i/x_j} \quad (4-11)$$

ReSiggen: 云服务器在收到代理重签名修改密钥后实施代理重签名。选择 $\omega \in Z_p$ ，计算：

$$\varphi'_i = (\phi_i^\omega, \varepsilon_i^\omega, \gamma_i^\omega) = (H(m)^{x_i\omega}, g^{x_i\omega}, g^\omega) \quad (4-12)$$

如果设 $\varpi = \omega \cdot x_i/x_j$ ，则可以将其转换为 $\varphi'_i = (H(m)^{x_j\varpi}, g^{x_j\varpi}, g^\omega)$ ，利用修改密钥 uk_j 计算 uk_j^ω ，并代替 g^ω 。将当前数据管理者 U_j 对应的签名验证信息记为：

$$\varphi_j = (\phi_j, \varepsilon_j, \gamma_j) = (H(m)^{x_j\varpi}, g^{x_j\varpi}, g^\varpi) \quad (4-13)$$

Verify: 当前数据管理者 U_j 通过以下两个式子来验证签名的有效性：

$$e(\phi_j, g) = e(H(m), \varepsilon_j) \quad (4-14)$$

$$e(\varepsilon_j, g) = e(g^{x_j}, \gamma_j) \quad (4-15)$$

4.3.3 PRRPA 方案

在本节，给出支持用户可撤换的云存储数据公开完整性验证方案：PRRPA，该方案是基于 4.3.2 小节的单向代理重签名技术构造的。见图 4-7，方案中定义了一个半可信的第三方审计者 TPA，它将忠实地执行数据完整性的完整性验证方案，由于它是好奇的，它可能会借助强大的计算设备从验证信息中通过解线性方程组恢复原始数据块信息，因此在方案中我们采用随机掩饰码技术来解决这一问题。

由于 PRRPA 方案涉及到大量的符号，为方便阅读，在表 4-3 中对所用符号进行列表说明：

方案中系统初始状态包括用户 U_1, U_2, \dots, U_m , 对应数据管理周期 T_1, T_2, \dots, T_m 。方案由以下七个算法组成: $Setup, SigGen, ReKeygen, ReSiggen, Challeng, ProofGen, ProofVerify$ 。

令 G 和 G_T 是两个生成元为 g 的 p 阶素数循环群。令 $e : G \times G \rightarrow G_T$ 表示双线性映射, $H : \{0, 1\}^* \rightarrow G$, $h : \{0, 1\}^* \rightarrow G$ 为安全的哈希函数, $f_{k_3} : \{0, 1\}^* \rightarrow Z_p$ 为伪随机函数。这里 k_3 仅为第三方审计者和云服务器所拥有。

算法 1. 初始化 $Setup$: 算法输入安全参数 λ , 对于每一个用户 U_j , $j \in \{0, \dots, m\}$ 有:

1. 选择一个随机数 $x_j \in Z_p$
2. 计算其公钥 g^{x_j}
3. 输出公钥 $pk_j = g^{x_j}$ 和私钥 $sk_j = x_j$

算法 2. 生成原始标签 $SigGen$: 输入初始用户 U_0 的公私钥对 (g^{x_0}, x_0) , 文件 $F \in \{0, 1\}^*$ 。

1. 初始用户 U_0 将文件 F 分成 n 块, 这里有 $m_i \in Z_p$, i.e., $(m_1, \dots, m_n) \in (Z_p)^n$ 。
2. 对于所有的 $i \in \{1, \dots, n\}$ 计算数据块 W_i 的标签为 $\sigma_i^{(0)} = (H(W_i) u^{m_i})^{x_0}$ (注: 文中使用 $\sigma_i^{(j)}$ 代表数据块 W_i 的标签由用户 U_j 的私钥 x_j 签名生成), 其中 u 是从 G 中选取的公开参数, 对应数据管理周期 T_1, T_2, \dots, T_m , T_j 是周期唯一标识, $T_0 = 1$,
计算 $t_j = T_j || Sig_{ssk}(T_j)$ 。
3. 将初始验证数据元 $V = \{\sigma_i, t_j\}_{1 \leq i \leq n, 1 \leq j \leq m}$ 和文件 F 传给云服务器, 然后再在本地删除。

算法 3. 修改密钥 $ReKeygen$: 如果用户 U_{j+1} 将取代用户 U_j , 他需要计算修改密钥 $uk_{j \rightarrow j+1}$ 并最终将它发送到云服务器。

$$uk_{j \rightarrow j+1} = (g^{x_j})^{\frac{1}{x_{j+1}}} = g^{x_j/x_{j+1}} \quad (4-16)$$

算法 4. 代理重签名 $ReSiggen$: 当每一个用户的管理周期结束, 云服务器接收到新用户的修改密钥执行代理重签名。

1. 设 $\alpha_j = pk_j = g^{x_j}$, $\beta_j = uk_{j \rightarrow j+1} = g^{x_j/x_{j+1}}$, $\omega_j = \prod_{\ell \in [1, j]} h(T_\ell)$, 其中 $h(T_j) \in Z_p$ 是一个代表周期 T_j 的哈希值。若记 $\tau_j = h(T_j)^{\frac{x_{j-1}}{x_j}}$, $\varpi_j = \prod_{\ell \in [1, j]} \tau_\ell$, 则有 $\varpi_j = \omega_j \frac{x_0}{x_j}$ 。

2. 云服务器在每次收到修改密钥后对数据块 m_i 实施一次重签名, 其中第 $j-1$ 层到第 j 层的重签名为: $\phi_j = (\phi_{j-1})^{h(T_j)} = (\sigma_i^{(0)})^{\omega_j} = (\sigma_i^{(j)})^{\varpi_j}$ 。这里表明新标签 ϕ_j 可由 ϕ_{j-1} 和当前周期 T_j 的哈希值 $h(T_j)$ 生成, 并归结为初始用户数据验证标签 $\sigma_i^{(0)}$ 的 j 次代理重签名, 并最终转换为当前用户 U_j 的重签名标签 $\sigma_i^{(j)}$ 的代理重签名形式。

表 4-3：符号说明表

符号	表达意义
U_0	系统初始用户，负责对初始数据块进行签名并将数据上传云服务器
U_1, U_2, \dots, U_m	用户负责管理数据，是动态可撤销的
T_1, T_2, \dots, T_m	用户对应管理周期的唯一标识
U_j	当前用户，其对应的管理周期为 T_j
$H(W_i)$	，是数据块序号与数据周期经过哈希运算生成的标识
$\sigma_i^{(j)}$	代表由用户 U_i 的私钥 x_i 签名生成的数据块 的数据标签，其中 $\sigma_i^{(0)}$ 为初始标签
$Sig_{ssk}(T_j)$	表示对周期序号的签名
t_j	$t_j = T_j Sig_{ssk}(T_j)$ ，为了防止 T_j 被修改而由用户生成的标记
$uk_{j \rightarrow j+1}$	修改密钥，如果用户 U_{j+1} 取代用户 U_j ， U_{j+1} 必须使用自己的私钥和 U_j 的公钥计算修改密钥
α_j	是为了表示方便使用的一组中间参数， $\alpha_j = pk_j = g^{x_j}$
β_j	是为了表示方便使用的一组中间参数， $\beta_j = uk_{j \rightarrow j+1} = g^{x_j/x_{j+1}}$
$h(T_j) \in Z_p$	一个代表周期 T_j 的哈希值，由用户生成，用于云服务器对数据标签进行处理。每一层的签名转换都要使用一个对应的 $h(T_j) \in Z_p$
τ_j	$\tau_j = h(T_j)^{\frac{x_{j-1}}{x_j}}$ 是一组中间参数。 τ_j 实际上是不存在的，因为云服务器不可能知道用户的私钥，但是通过代理重签名的多层迭代，可以相互抵消，最终得到一个已知的参数
ω_j	$\omega_j = \prod_{\ell \in [1, j]} h(T_\ell)$ 是一组中间参数。用来表示当前签名转换过程中所使用到的 $h(T_j) \in Z_p$ 的连乘，云服务器对数据标签以及一些相关参数的重签名一开始表示成以 ω_j 为指数的幂形式
ϖ_j	$\varpi_j = \prod_{\ell \in [1, j]} \tau_\ell$ 是为了表示方而使用的一组中间参数。用来表示当前签名转过程中所使用到的 τ_j 的迭代情况。 $\varpi_j = \omega_j^{\frac{x_0}{x_j}}$ 揭示了 $\omega_j = \prod_{\ell \in [1, j]} h(T_\ell)$ 与 ϖ_j 之间的转换关系。云服务器对数据标签以及一些相关参数的签名最终由 ω_j 为指数的幂的形式转换成 ϖ_j 为指数幂的形式
ϕ_j	$\phi_j = (\phi_{j-1})^{h(T_j)} = (\sigma_i^{(0)})^{\omega_j} = (\sigma_i^{(j)})^{\varpi_j}$ 表示数据块 对应数据标签经多层转换后的形式；
ε_ℓ	$\varepsilon_\ell = (\alpha_{\ell-1})^{\prod_{k \in [\ell, j]} h(T_k)} = (\alpha_j)^{\prod_{k \in [\ell, j]} \tau_k}$ 用来表示用户公钥多层转换后的形式。最终经过多层转换后的已撤销用户的公钥集合将作为最终审计证明响应信息的一部分
γ_ℓ	$\gamma_\ell = (\beta_\ell)^{h(T_\ell)} = g^{h(T_\ell)^{\frac{x_{\ell-1}}{x_\ell}}} = g^{\tau_\ell}$ 用来表示用修改钥转换后的形式。最终经过多层转换后的用户修改密钥集合将作为最终审计证明响应信息的一部分
φ_j	$\varphi_j = (\phi_j, t, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 用来表示第 次重签后 m_i 的完整性验证响应信息
V_j	$V_j = (\mu, \phi, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 用来表示最终审计证明响应信息
Q	$Q = \{(i, v_i)\}$ 中 i 是数据块的序号，而 v_i 是随机数， Q 是随机选取的一系列 i 和 v_i 的集合。用来指定将要挑战的数据块和数据块序号
	为挑战数据块所对应的标签最终转换形式的最终聚合状态表示

3. 云服务器在对数据进行重签名的同时，对用户公钥和重签名修改密钥进行处理，对于 $\ell \in (1, \dots, j)$ 计算：

$$\begin{aligned}\varepsilon_\ell &= (\alpha_{\ell-1})^{\prod_{k \in [\ell, j]} h(T_k)} = (\alpha_j)^{\prod_{k \in [\ell, j]} \tau_k}, \\ \gamma_\ell &= (\beta_\ell)^{h(T_\ell)} = g^{h(T_\ell) \frac{x_{\ell-1}}{x_\ell}} = g^{\tau_\ell},\end{aligned}$$

当 $\ell = j$ 时有

$$\varepsilon_j = (\alpha_{j-1})^{h(T_j)} = (\alpha_j)^{h(T_j) \frac{x_{j-1}}{x_j}} = (\alpha_j)^{\tau_j}, \quad \gamma_j = (\beta_j)^{h(T_j)} = g^{h(T_j) \frac{x_{j-1}}{x_j}} = g^{\tau_j}.$$

最后将 $\varphi_j = (\phi_j, t, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 作为第 j 次重签后 m_i 的签名验证信息。

在后面的完整性验证过程中当第一次 TPA 收到来自云服务器的，可以通过用户的公钥计算如下：

$$PK_{(U_j)}(Sig_{ssk(U_j)}(T_j)) = T_j \quad (4-17)$$

来验证 T_j 的正确性。并利用哈希函数 $h : \{0, 1\}^* \rightarrow G$ 计算 $h(T_j)$ 。出于简捷性考虑，我们假设 TPA 已知 T_j ，在后面的验证不再考虑验证 t_j ，并不计入通信和计算复杂度。

算法 5. 挑战Challeng: 如果当前用户 U_j 想验证存储在云服务器上的数据块的完整性，他可以发送一个request给 TPA。当 TPA 收到这个请求后生成随机的一系列 $Q = \{(i, v_i)\}$ （这里 $i \in \{1, \dots, n\}$, $v_i \in Z_p$ ），和一个通信密钥，并发送给云服务器作为一个挑战Challeng。

算法 6. 证据生成ProofGen: 当云服务器收到来自 TPA 的挑战Challeng，它将计算：

$r = f_{k_3}(challenge) \in Z_p$, $R = u^r \in G$, $\mu' = \sum_{i \in Q} v_i m_i$, $\mu = \mu' + rh(R) \in Z_p$ 。
得到审计证明响应信息 $V_j = (\mu, \phi, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 并返给 TPA。

算法 7. 证据验证ProofVerify: 当 TPA 收到来自云服务器的审计证明响应信息 $proof$ 。运行下面的算法，当验证通过时输出VALID否则输出INVALID。

$$e(\phi, g) = e(u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q} H(W_i)^{v_i}, \varepsilon_1) \quad (4-18)$$

$$e(\varepsilon_j, g) = (\alpha_j, \gamma_j) \quad (4-19)$$

$$e(\varepsilon_\ell, g) = e(\varepsilon_{\ell+1}, \gamma_\ell) \quad for \ell \in \{0, \dots, j-1\} \quad (4-20)$$

4.3.4 PRRPA 方案的正确性与安全性分析

4.3.4.1 正确性

根据前面的内容，我们可以看到对于任何挑战Challeng, $Q = \{(i, v_i)\}$ 、 有：

$$\begin{aligned}
 e(\phi, g) &= e\left(\left(\prod_{i \in Q} \sigma^{(j)}\right)^{v_i \varpi_j}, g\right) \\
 &= e\left(u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q} H(W_i)^{v_i}, \alpha_j^{\varpi_j}\right) \\
 &= e\left(u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q} H(W_i)^{v_i}, \varepsilon_1\right)
 \end{aligned} \tag{4-21}$$

对于任意的用户 U_j , $j \in \{1, \dots, m\}$ 有:

$$\begin{aligned}
 e(\varepsilon_j, g) &= e((\alpha_j)^{\tau_j}, g) \\
 &= (\alpha_j, g^{\tau_j}) = (\alpha_j, (\beta_j)^{h(T_j)}) = (\alpha_j, \gamma_j)
 \end{aligned} \tag{4-22}$$

对于所有的 $\ell \in \{1, \dots, j-1\}$ 有:

$$\begin{aligned}
 e(\varepsilon_\ell, g) &= e(g, g)^{x_j \prod_{k \in [\ell, j]} \tau_k} = e\left(g^{x_j \prod_{k \in [\ell+1, j]} \tau_k}, g^{\tau_\ell}\right) \\
 &= e(\varepsilon_{\ell+1}, \gamma_\ell)
 \end{aligned} \tag{4-23}$$

因此, 我们的方案是正确的。

4.3.4.2 安全性分析

下面我们首先证明该方案能够抵抗恶意云服务器通过产生伪造的审计证明响应信息来欺骗 TPA 的完整性验证过程。

定理 4.4: 恶意的云服务器产生伪造的审计证明响应信息 $Proof$ 来欺骗 TPA 的完整性验证过程在计算上是不可行的。

证明: 假设存在恶意的云服务器 (多项式时间对手 A) 以不可忽略的概率 ξ 产生伪造的审计证明响应信息来欺骗 TPA 的完整性验证过程。下面我们设置多项式时间算法 (挑战者 B) 通过运行对手 A 作为子程序也以不可忽略的概率 ξ 解决 CDH 困难题。算法 B 与对手 A 的信息交互如下:

Setup. 给定一个安全参数 λ , 算法 B 首先随机选取 G 的一个生成元 g , 这里 $g^\alpha \in G$, $H : \{0, 1\}^* \rightarrow G$ 将在证据里被模式化为随机预言模型。对于未知的 x_0 , B 也可以随机的从 Z_q 中选取 g^{x_0} 作为用户 U_0 的公钥, 同样对于 $j \in \{1, \dots, m\}$, 从 Z_q 中选取 x_j 计算用户 U_j 的公钥 g^{x_j} 。然后 B 设置 $u = g^\alpha$, 然后将系统参数 g , u 和所有的用户公钥 $\{g^{x_j}\}_0^m$ 发送给对手 A 。

Query. 对手 A 可以询问以下自适应模型。我们假设, 对任意数据块 m_i , 首先做一个 $H - Oracle$ 查询。

$H - Oracle$: 当 A 向预言机询问数据块 m_i , B 在 H -list 中查询 m_i 对应的元组 $(m_i, s_i, H(W_i))$ 。如果 B 找到了一个匹配对, 他输出响应 $H(W_i)$ 。否则 B 首先选择一个随机值 $s_i \in Z_p$ 然后计算 $H(W_i) = g^{s_i}/u^{m_i}$, 将 $(m_i, s_i, H(W_i))$ 存储于 H -list 最后输出响应 $H(W_i)$ 。同样 A 在 H -list 中查询 T_j 对应的输出 $h(T_j)$ 。

$SignGen - Oracle$: A 向预言机询问获取数据块 $\{m_i\}_{i \in [1, n]}$ 的标签。一旦收到

询问对于 $i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$, $\ell \in \{0, \dots, j\}$, B 在 H-list 中查询 m_i 寻找匹配元组并计算 $\sigma_i = (g^{x_0})^{s_i}$ 最终输出一系列 $V = (\sigma_1, \dots, \sigma_n)$ 作为响应。因为将 $\sigma_i = (H(W_i) u^{m_i})^{x_0}$ 代入等式得到 $H(W_i) = g^{s_i} / u^{m_i}$ 。我们可以得到对于 $i \in \{1, \dots, n\}$, $H(W_i) = g^{s_i} / u^{m_i}$ 。

Update-Oracle: 如果 A 想用继任者 U_{j+1} 来代替用户 U_j , $j \in \{1, \dots, m-1\}$, A 将询问 U_j 。一旦收到询问, B 首先利用 U_{j+1} 的私钥 x_{j+1} 计算修改密钥 $uk_{j \rightarrow j+1} = g^{x_j/x_{j+1}}$ 。并将结果反馈给 A 。

ReSiggen-Oracle: 然后 A 设置 $\alpha_j = g^{x_j}$ 和 $\beta_{j+1} = uk_{j \rightarrow j+1}$, 则对于 $\varpi_j = \omega_j^{\frac{x_0}{x_j}}$, $\tau_j = h(T_{U_j})^{\frac{x_{j+1}}{x_j}}$ 有 $\phi_j = (\sigma^{(0)})^{\varpi_j}$, $\varepsilon_\ell = (\alpha_j)^{\prod_{k \in [\ell, j]} \tau_k}$, $\gamma_\ell = g^{\tau_\ell}$ 。并将他们加入 U_j 的验证元组。让 $\varphi_j = (\phi_j, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 作为 U_j 验证元组。

Corrupt-Oracle: 设置所有的已撤销用户为 U_1, \dots, U_d , 这里 $d \in \{1, \dots, m-1\}$ 。如果对手针对 U_j 询问预言机, 这里 $j \in \{1, \dots, d\}$, B 就返回 U_j 的私钥 x_j 作为响应。

Proof: 如果 B 想验证数据 m 是否仍然完整地存储在云上, 他将生成一个随机挑战 $Challeng(Q, k_3)$, 其中 $Q = \{(i, v_i)\}$ 给 A , 这里 $i \in \{1, \dots, n\}$, $v_i \in Z_p$ 。令 $(m_1, \dots, m_n) \in (Z_p)^n$ 当前用户为 U_j 。一旦收到 $Challeng(Q, k_3)$, A 计算 $r = f_{k_3}(challenge) \in Z_p$, $R = u^r \in G$, $\mu' = \sum_{i \in Q} v_i m_i$, $\mu = \mu' + rh(R) \in Z_p$ 、 $\phi = (\prod_{i \in Q} \sigma_i^{v_i})^{\varpi_j}$ 并生成一个有效的证明 $V_j = (\mu, t, \phi, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 给 B 。

Forgery: A 以不可忽略的概率 ξ 输出有效证据 $(\mu^*, \phi^*, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 作为用户 U_j , $j \in \{1, \dots, m\}$ 对已损坏数据块 $\{m_i\}_{i \in [1, n]}$ 的挑战的 $Challeng(Q, k_3)$ 的响应。令数据块未损坏的情况下原本 U_j 挑战 $\{m_i\}_{i \in [1, n]}$ 响应 $Challeng(Q, k_3)$ 的输出证明为 (μ, σ) , 这里显然 $\mu \neq \mu^*$ 。令 $\Delta\mu = \mu^* - \mu$ 由于

$$e(\phi, g) = e\left(\varepsilon_1, u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q} H(W_i)^{v_i}\right) \quad (4-24)$$

$$e(\phi^*, g) = e\left(\varepsilon_1, u^{\mu^*} \cdot R^{-h(R)} \cdot \prod_{i \in Q} H(W_i)^{v_i}\right) \quad (4-25)$$

可以得到 $e(\sigma^* \cdot \sigma^{-1}, g) = e(g^{x_0 \omega_j}, u^{\Delta\mu})$ 。由此结果我们可以从中推导出 $w^{x_0 \omega_j} = g^{ax_0 \omega_j} = (\sigma^* \cdot \sigma^{-1})^{\frac{1}{\Delta\mu}}$ 。由此, 给定 $g, g^a, g^{x_0} \in G$, 输出 g^{ax_0} , B 将以不可忽略的概率 ξ 解决 CDH 问题。■

私钥泄露问题: 根据上面定理 4.4 的证明过程, 可以轻易发现即使云服务器与已撤销用户合谋, 云服务器也不能够获取当前用户的私钥。这是因为方案中修改密钥的计算是当前用户利用自身的私钥通过对已撤销用户的公钥进行处理得到: $uk_{j \rightarrow j+1} = g^{x_j/x_{j+1}}$ 。如果云服务与已撤销用户想通过合谋获取当前用户私钥, 就必须先以不可忽略的概率解决 DL 问题, 再通过已撤销用户的私钥计算出当前用户私

钥。而文献[49]中 $uk_{j \rightarrow j+1} = x_{j+1}/x_j$, 如果云服务与已撤销用户合谋, 那么云服务器很容易就能获取当前用户私钥。

下面我们证明方案满足隐私保护性:

定理 4.5: 给定一个来自云服务器的审计证明响应信息 $proof$, 对于好奇的 TPA, 它试图从中恢复用户数据文件 $F = \{m_1, m_2, \dots, m_n\}$ 中的数据块是不可行的。

证明: 组合信息 $\mu' = \sum_{i \in Q} v_i m_i$ 是关于用户原始数据块的线性组合, 一旦这个组合信息发送给 TPA, 这个好奇的 TPA 可以通过收集大量的组合信息, 并借助强大的计算设备来求解这些线性方程组, 从而恢复用户的原始数据块。为了防止 TPA 读取用户的原始数据块信息, $\mu' = \sum_{i \in Q} v_i m_i$ 需要使用随机掩饰码技术, 具体如下:

云服务器利用伪随机函数 f 计算随机值 $r = f_{k_3}(challenge) \in Z_p$, 并计算 $R = u^r \in G$, 最后将 $\mu' = \sum_{i \in Q} v_i m_i$ 盲化 $\mu = \mu' + rh(R) \in Z_p$ 这样为了让 TPA 仍然能解这些线性方程, TPA 必须掌握这个随机值 r , 进而它必须掌握这个伪随机函数的秘密密钥, 事实上, 这个秘密密钥只有云服务器知道, 因此 TPA 不可能知道这个随机值 r 。因此, 对于好奇的 TPA, 它试图从审计证明响应信息恢复用户数据文件是不可行的。■

4.3.5 PRRPA 方案的性能分析

在本节分析 PRRPA 方案的通信和计算复杂度, 并与文献[49]中的 Panda 方案和前文中的 EDRPA 方案作比较。为统一起见, 与前文一致只计算频繁完整性验证过程中的通信和计算代价而不计算系统建立时的通信与计算代价。

表 4-4 三个方案的效率比较

方案	通信开销	计算开销		合谋抵抗	公开审计
		修改密钥	验证时间		
Panda 方案 ^[49]	$d \cdot (p + G) + c \cdot (id + n + p)$	$nExp$	$(c + 2d) MZ + dMG + (c + d) Exp + (d + 1) Pair$	不	是
EDRPA 方案	$ p + G + c(2 C + p)$	Exp	$(c + 2d) MZ + jMG + (2c + d) Exp + (d + 1) Pair$	是	是
PRRPA 方案	$c n + (c + 1) p + G $	Exp	$(c + 1) MZ + MG + (c + 3) Exp + 2Pair$	是	是

我们用 $Pair$ 表示双线性对操作, Exp 表示 G 上的模指数操作, MZ 和 MG 分别表示 Z_p 和 G 上的乘法操作。 $|n|$ 、 $|p|$ 、 $|G|$ 分别表示 $\{1, \dots, n\}$ 、 Z_p 和 G 的比特长度。挑战中选取的数据块假定是个常量 c , 挑战中已撤换的用户数量总和也假设是个常量 d 。

1. **通信开销:** 可以看到 PRRPA 方案中通信负载主要取决于审计证明响应信

息的通信代价。其中发送挑战 $Q = \{(i, v_i)\}$ 到云服务器的通信量为 $c(|n| + |p|)$ 。云服务器返回审计证明响应信息 $V_j = (\mu, \phi, \{\varepsilon_\ell\}_{\ell \in [1, j]}, \{\gamma_\ell\}_{\ell \in [1, j]})$ 的通信量为 $|p| + (2d + 1)|G|$, 但是注意到只有在用户 U_j 发起的第一次完整性验证方案请求中才需要这样的通信量, 否则只需要传送通信量为 $|p| + |G|$ 的 (μ, ϕ) 就行了。因此最终在一次完整性验证过程中总共的通信开销为 $c|n| + (c + 1)|p| + |G|$ 。

2. 计算开销: 计算开销包括修改密钥生成、代理重签名和验证开销三部分, 由于本方案支持第三方公开完整性验证方案, 所以只需要考虑修改密钥生成和验证开销两部分。对于用户 U_j , 在 $ReKeygen$ 算法只需要计算 g^{x_{j-1}/x_j} , 因此计算代价为 Exp 。根据上面提到的计算原则, 并简化加操作和哈希操作后我们计算出验证开销为: $(c + 1)MZ + MG + (c + 3)Exp + 2Pair$ 。

3. 开销比较: 如表 4-4 所示对比 Wang 等人[49]中 Panda 方案与前文 EDRPA 方案, PRRPA 方案通信开销与计算开销皆优于上述两个方案。

4.4 本章小结

本章聚焦于支持用户可撤销的云存储数据完整性验证性问题。首先分析了支持用户可撤销的云存储数据完整性验证的系统模型、方案构成以及安全性需求。接着基于双线性对聚合签名技术提出了一个有效和安全的公开完整性验证方案 EDRPA。然后, 进一步深化, 基于单向代理重签名技术提出了一个具有隐私保护的支持用户可撤销的云存储数据公开完整性验证新方案 PRRPA。最后, 文中对提出的两个方案进行了性能分析, 分析表明两个方案均在增加新功能的基础上, 对比 Panda 方案验证效率仍然具有优势, 其中以 PRRPA 方案最优。

第五章 支持密钥更新的高效云存储数据完整性验证方案

本章研究的主要内容是支持密钥更新系统中高效的云存储数据完整性验证问题。在嵌入式云存储应用背景下，嵌入式用户可以同时享受移动网络和云存储带来的双重便利。但数据的完整性仍然是嵌入式用户在嵌入式云存储中的主要担忧。为解决这个问题，一系列数据完整性验证方案被随之提出。然而，在用户没有对数据重新拥有的情况下，如何更新用户的完整性验证密钥（以及这些密钥相关的验证标签）仍然是一个有待解决的问题。同时，受限制于存储、计算、带宽等资源有限的应用背景，数据完整性方案的效率与密钥更新效率问题也必须被正视。在本章中，为嵌入式云数据完整性验证提供了一个存储文件零知识隐私保护下的密钥更新和验证标签更新的机制，这种机制由零知识证明系统、代理重签名和线性同态验证器组成。并通过改进 Shacham-Waters 完整性验证方案来实例化本章的提案。当嵌入式用户需要更新自己的密钥时，用户只需要下载并更新验证标签，而不需要下载整个文件和重新生成所有的验证标签。这种方法在保证理想安全状态的同时，显著的降低了通信和计算开销，是一种支持密钥更新与零知识隐私保护的实用嵌入式云存储数据公开完整性验证方案。

本章内容由六部分构成：5.1 节对嵌入式云存储系统数据完整性验证问题进行概述；5.2 节给出嵌入式云存储系统数据完整性验证的系统模型以及安全模型；5.3 节给出具体的支撑密钥更新与零知识隐私保护的实用嵌入式云存储公开完整性验证方案；5.4 给出安全性分析；5.5 对其进行性能分析；5.6 节对本章进行总结。

5.1 嵌入式云存储基本概述及相关研究工作

嵌入式系统^[111]已经广泛地应用于消费、商业、汽车、工业和医疗市场。据估计，截止 2015 年底，超过 150 亿的嵌入式设备将被连接到互联网上。因此，随着无线技术的发展嵌入式设备（尤其是各种移动嵌入式设备）已经成为了一种方便的普及式的互联网服务接入终端。然而，嵌入式设备的存储空间有限、计算能力弱、并受到蓄能能力的限制，这些都导致了它无法支持复杂的数据挖掘和大数据存储。云作为一种具有强大计算能力和存储容量的丰富资源，它的出现可以扭转这个状况，并能极大地扩展嵌入式设备的功能^[112]。通过云计算，嵌入式设备由于受到海量的存储数据和潜在的数据分析能力的影响，嵌入式设备将会变得越来越智能化^[1]。能够从互联网范围获取智力支持，这将使得对它的企业价值发生深刻变化。例如：微软营销总监声称在嵌入式操作系统（Windows Embedded）中，云将

在扩展嵌入式计算方面起到关键作用。微软目前正致力于开发一种软件，其目的是实现集成嵌入式设备和云之间的重要功能。

作为嵌入式云计算的主要服务领域之一，嵌入式云存储允许嵌入式用户将数据存储在云上，以弥补嵌入式设备存储容量的不足。这种新的存储服务有许多优点，比如减轻嵌入式用户数据管理和维护的负担，避免经费开销集中在软硬件上^[113]。然而，尽管嵌入式云存储具有如此吸引人的特点，但同时也带来了一系列的安全性挑战问题^[6]。由于失去了对外包数据的物理占有和控制，嵌入式用户会担心他们的数据可能由于以下原因被篡改或删除。首先，不管云服务供应商提供多么可靠的可靠性措施，仍然会因基础设施问题造成不可挽回的数据损坏。其次，在云存储中，云存储服务提供商并不值得完全信任。云服务提供商可能会因为经济原因丢弃很少访问的数据，但对外声称数据仍然被很好地存储，或隐藏数据丢失事件以保持其声誉。云安全联盟（CSA）对诸如停电、停机、数据丢失等云计算的弱点进行了系统的调查报告。CSA 在 2013 年发布的一份白皮书中揭示首要的三大威胁“不安全的接口和 API”、“数据丢失和泄漏”和“硬件故障”。这三个威胁占所有云中断事件的 64%，而“数据丢失和泄漏”占 25%。因此，对于一个安全的嵌入式云存储来说，在云计算中保证数据的完整性或对数据进行完整性验证具有很高要求。尽管到目前为止有很多云数据完整性验证方案^[16,26,29,38,41,43-44,114]已经被提出来，但是它们都是在传统的云存储环境下设计的，而没有考虑嵌入式云存储的应用。

到目前为止，已经有两类数据完整性验证方案：即基于公钥基础设施（PKI）机制的和基于伪随机函数（PRF）机制。基于伪随机函数（PRF）的方案是自身可验证的，这意味着只有数据拥有者可以检查外存数据的完整性。作为比较，公开验证的完整性验证方案允许外部第三方审计者在需要时验证数据的完整性。嵌入式设备通常被资源约束且不支持复杂性计算。对于嵌入式用户来说，能够使用嵌入式云存储公开完整性验证方案是至关重要的。在本章中，只考虑公开验证的完整性验证方案，因为这些方案在实际中应用非常广泛。实现公开验证的关键技术是 Ateniese 等人在文献[24]中提出的同态认证。

在 PKI 系统中，要用到认证中心（CA），CA 的主要作用是进行数字签名和对认证用户颁发证书。公钥密码体制使用证书来处理角色问题。证书是一种用于在指定有效期内将个人、公司或其他的任何实体的公钥相关联的电子文件。PKI 中的关键问题是如何处理用户密钥泄漏或失效的问题。密钥过期表明用户证书不再有效，而密钥泄漏将给合法用户带来严重的安全威胁。因此，在实际应用中，证书撤销和更新补发是维护互联网通信安全支撑 PKI 安全体系的关键环节。当私钥泄

漏后，证书必须被撤销，因为该证书的拥有者已经不能再控制该证书的使用，或者该证书可能会生成错误的签名。如果 CA 没有撤销证书的能力，它就不能够在证书到期之前将之标记为不受信任的证书，而证书到期往往可能需要几年的时间。在嵌入式的云存储中，当用户出于各种原因需要改变他的密钥时，会出现一个新的困难问题，如何验证云存储的外包数据的完整性。因为将数据存储在云端的原验证者在密钥更新后是不再有效的。面对这个问题，一个简单的解决方案是从云端下载所有数据块和验证标签，重新计算每个数据块的验证标签并上传至云端。这种模式将同时给嵌入式用户和云服务器带来不可接受的通信成本，同时也为嵌入式用户带来了显著的计算成本。为解决这个问题，在数据完整性验证方案中采用高效的密钥更新和认证协议对嵌入式云存储的实用化来说是必不可少的。

嵌入式云数据完整性验证方案中的另一个重要问题是数据的隐私保护。也就是说，完整性验证过程中不应该向第三方审计者透露挑战文件的内容。注意到一些嵌入式设备，如智能手机、智能摄像头、苹果手表和运动手环等，可能会产生大量的个人私密数据信息。如果这些敏感信息在完整性验证过程中暴露出来，嵌入式用户的隐私比如其社会关系、物理状态和家庭背景可能会被泄露给完整性验证者或公众。

这种信息泄露可能有助于黑客给嵌入式用户带来灾难性后果。因此，在数据完整性验证过程中，嵌入式用户不愿意透露任何信息给第三方审计者，所以第三方审计者唯一的工作就应是检查外包存储文件的完整性。早期的完整性验证方案不具有隐私保护性。应对挑战，云服务器产生 $u_i = \sum_{(i, v_i)} v_i m_i$ 作为响应，（其中 m_i 是被挑战的数据块而 v_i 是由挑战中验证者生成的随机数）。由于，每一个 u_i 都透露了数据块的部分信息，通过重复地挑战这些数据块，第三方审计者就可以得到这些数据。Wang 等在 2010 年提出了第一个隐私保护的公开完整性验证方案^[28]。他们通过盲化采样文件块的线性组合改进了 Shacham-Waters 方案，使得第三方审计者无法从云服务器发送的审计证明响应信息中恢复出文件块。不幸的是，Xu 等人在文献[35]中发现该协议容易受到恶意云服务器和外部攻击者的攻击。随后，Wang 等人改进了他们的方案[26]以使得它对于来自外部的伪造攻击变得更加安全。然而，我们认为在一些实际应用中第三方审计者无法从整个文件块中获取信息还是不够的。例如，第三方审计者可以发起一个离线猜测攻击，从若干存储在云服务器中文件中获取文件。Wang 等人提出了抵御离线猜测攻击的“零知识公共完整性验证方案”的概念。然而，非常关键的安全模型并未在他们的论文中被提出。Yu 等人^[115]最近改进了安全云存储的远程数据完整性检查协议，但是他们的模型中并不支持密钥更新场景。

在本章中，设计了基于零知识数据隐私技术的数据完整性验证方案，并提出了一种技术有效地解决了密钥更新问题。详细地说，本章利用简单的离散对数零知识证明使得 Shacham-Waters 协议^[25]在密钥更新环境中不会泄漏任何外包数据的内容信息给验证者。同时，当嵌入式用户的密钥更新或到期时，不需要再从云端下载他的数据，而是从云端下载一个与整个文件相比非常小的认证作为替代。因此，用户可以有效地改变验证标签，从而大大地降低了通信代价和计算开销。方案同时证明了合理的新安全模型在随机预言模型下的是可证明安全的。最后，通过开发一个原型来评估方案框架的性能。实验结果与理论分析相吻合，表明新协议是实用的。

5.2 系统模型

在本节中，回顾了可进行公开完整性验证的嵌入式云存储系统模型，并定义了密钥更新数据完整性验证方案在嵌入式云存储中的安全模型。

5.2.1 嵌入式云存储数据完整性验证系统模型

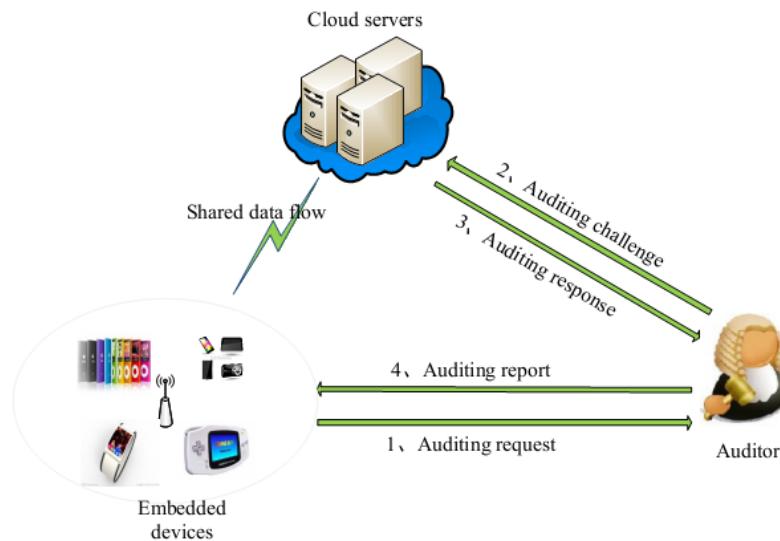


图 5-1 嵌入式云存储数据完整性验证方案系统模型

一个可公开验证的嵌入式云数据完整性验证架构^[24-25]如图 5-1 所示。其应用场景涉及三个实体：嵌入式用户、云服务器和第三方审计者（TPA）。嵌入式用户生成数据文件，并将大量数据存储在远程云服务器上，而不需要再要保存本地副本。云服务器具有很大的存储空间和丰富的计算资源，为嵌入式用户提供数据存储服务。TPA 是一个可由政府管理的可信组织，具有云用户所不具备的专业知识和能

力，其在嵌入式用户的请求下代表用户去检查托管数据完整性的行为是可以信赖的。每个实体都有自己的义务和利益。云服务器可能是自私的，比如为了维护自己的声誉，云服务器甚至可能会向其他人隐藏数据损坏事故。更进一步，在当前周期云服务器可能尝试获取数据所有者在先前周期的密钥。尽管如此，仍假设云服务器没有法律和财务动机向 TPA 泄露托管数据。一些嵌入式用户可能没有时间、资源或可能性去监控他们数据，TPA 将负责代表用户检验云数据的完整性并向用户返回完整性验证报告。在零知识隐私保护的完整性验证方案中，TPA 在完整性验证方案过程不能获得存储数据的任何信息。

5.2.2 方案定义及其安全性

密钥更新的公共数据完整性验证方案由以下四个算法组成：($CrsGen$, $KeyGen$, $AuthGen$, $AuthUpdate$)。

$CrsGen(1^k)$: 该算法输入一个安全参数 k ，输出一个公共参数 crs （假设算法结束时，其他各方都已经得到 crs ，这对于以下描述的算法来说是一个隐式输入）。

$KeyGen(crs)$: 输入 crs ，这个算法为嵌入式用户生成一个公钥 pk 和一个私钥 sk ，用户发布 pk 并秘密保存 sk ，注意，这个算法同样被用于密钥更新。

$AuthGen(sk, F)$: 输入私钥 sk 和一个文件 $F = (m_1, \dots, m_n)$ ，输出这个文件的一个验证者集合 $\{D_i\}$ 和一组在证明过程中检验数据完整性的公开验证参数集合 ϕ 。

$Proof(P, V)$: 这是一个证明者(P)和验证者(V)之间的交互式协议，对(P, V)共同的输入是公钥 pk 和公开验证参数 ϕ ， P 有一个附加的输入文件 $F = (m_1, \dots, m_n)$ 和这个文件相应的验证者集合 $\{D_i\}$ ，在这个协议的结尾， V 输出一位 1 或 0 表明这个文件是否保存完整，为方便起见，使用 $P \Leftrightarrow V(pk, \phi) = 1$ 表示在与 P 交互的结尾 V 输出为 1。当上下文明确时省略参数 (pk, ϕ) 。

$AuthUpdate(sk, pk, sk', pk', \{D_i\}, \phi)$: 输入一个新的密钥对 (sk', pk') 和在旧的密钥对 (pk, sk) 下有效的一个验证者集合 $\{D_i\}$ 和公开验证参数 ϕ ，这个算法输出一个已演化的验证者集合 $\{D'_i\}$ 和一个在新的密钥对下有效的参数 ϕ' 。

5.2.2.1 正确性

正确性，可靠性、零知识数据隐私保护、高效性是本章所提出的隐私保护云存储数据公开完整性验证策略的四个基本要求。正确性意味着在与云服务器交互过程中如果云服务器存储数据没有被更改，只要云服务器和 TPA 诚实地执行协议，那么交互协议 $Proof$ 会始终得到结果 $P \Leftrightarrow V = 1$ 。

5.2.2.2 可靠性

可靠性是说任何证明者能够使验证者确信他的存储文件确实是那个存储文件本身。在下面的内容中，为支持密钥更新的云数据公开完整性验证，方案定义了基于 Shacham-Waters^[25]的安全模型以应对恶意服务器。这个模型中涉及两个实体，即一个对手代表不可信服务器和一个代表数据所有者或完整性验证者的挑战者。对手的目标是欺骗第三方审计者，即在不存储原始文件的情况下生成一个有效的证明响应信息。本章的模型和以往的 PDP 或 POR 模型之间的关键差别在于密钥更新和验证标签更新操作。

为描述可靠性，首先回顾在文献[25]中定义的一个概率界 ϵ —可接受证明定义。如果 $\Pr[P' \Leftrightarrow V(pk, \phi) = 1] \geq \epsilon$ ，那么对于 (pk, ϕ) 的算法 P' 是 ϵ 可接受的。也就是说，当一个可信的 TPA 运行 V 时，它能够令人信服的以 ϵ 的部分回应验证挑战。

考虑如下对手 A 和挑战者 C 之间的游戏：

Init. 挑战者 C 初始化了一个计数器 $cnt = 0$ 和一个空表 tbl ，它调用 $CrsGen$ 输入一个安全系数 k 以获得 crs 。然后调用 $KeyGen$ 去产生一个密钥对 (pk_{cnt}, sk_{cnt}) 。 crs, pk_{cnt} 被发送给对手 A 知道，假设 A 能够一直检索当前 cnt 和 tbl 的值。

AuthQuery. 假设一个文件 F 对应的 $AuthQuery$ 都是唯一的。万一 F 被重复提交， C 能够将存储在表 tbl 中的结果返回给 A 。 A 提交一个文件 $F = (m_1, \dots, m_n)$ ， C 利用输入 (sk_{cnt}, F) 运行 $AuthGen$ 算法计算相应的认证 $\{D_i\}$ 和公共验证参数 ϕ_F ， C 向表 tbl 中增加一行 $(F, \{D_i\}, \phi_F)$ ，如果 A 对数据表已经进行了一次访问，返回值被忽略。

ProofQuery. A 和 C 都参与了这个证明协议， C 将扮演 $TPA(V)$ 的角色，输入为 (pk_{cnt}, ϕ_F) 。当协议终止时 V 的输出将发送给 A 。

Update. 当 A 调用这个查询时， C 首先将设置 $cnt = cnt + 1$ 。然后， C 再次调用 $KeyGen$ 去获得一个新密钥对 (pk_{cnt}, sk_{cnt}) ，对于表 tbl 中每一行 $(F, \{D_i\}, \phi_F)$ ， C 调用 $AuthUpdate(sk_{cnt-1}, pk_{cnt-1}, sk_{cnt}, pk_{cnt}, \{D_i\}, \phi_F)$ 去获得 $(\{D'_i\}, \phi'_F)$ 。更新此列为 $(F, \{D'_i\}, \phi'_F)$ ，在这之后， C 再将 sk_{cnt-1} 返回给 A 。

Challenge. 在某些时候， A 输出一个证明算法 P^* 的描述。

首先回顾一下 $extractor$ （知识提取器）的概念，一个 ext 是概率界为 ϵ —可接受证明者 P' ，公钥 pk 和验证参数 ϕ ，输出一个文件 F ，表示为 $ext(P', pk, \phi) = F$ 。

现在定义可靠性。具体来说，定义所有的 PPT 算法 A ，输出一个满足 (pk_{cnt}, ϕ) 的 ϵ 可通过证明者 P^* ，存在一个淬取 ext ，如此 $(ext(P^*, pk_{cnt}, \phi), pk_{cnt}, \phi)$ 是表 tbl 中的一行。

换句话说，以某种格式存储原文件，当一个算法可以接受证明的概率比 ϵ 更大。则无论文件以任何格式存储都可以通过知识提取算法恢复出原始文件。

5.2.2.3 零知识数据隐私

零知识数据隐私的属性意味着 TPA 除了获取一个基于公开可用信息的存储文件的随机文件名外，将不会获取有关任何内容的信息。为了进一步增强这个属性，允许 TPA 选择两个相同长度的文件 $F_0 = (m_{0,1}, \dots, m_{0,n})$ 和 $F_1 = (m_{1,1}, \dots, m_{1,n})$ ，按要求给出一组数据元并与云服务器进行交互，TPA 并不能获得关于文件 F_0 和 F_1 的内容的任何信息。一个正式的安全模型描述如下。

对手 A 和挑战者 C 之间的游戏：

Init. 挑战者 C 初始化两个计数器 $cnt = 0, j = 1$ 和一个空表 tbl ，它调用 $CrsGen$ 算法，输入一个安全参数 k 以得到 crs 。然后调用 $KeyGen$ 算法产生一个密钥对 (pk_{cnt}, sk_{cnt}) 。 crs, pk_{cnt} 发送给对手 A ，假设 A 能够一直获取当前 cnt 和 tbl 的值。 C 初始化一个空行 $R^* = (0, \perp, \perp, \perp)$ ，被称之为挑战行。

AuthQuery(F) 假设 F 对于每个 $AuthQuery$ 都是唯一的。万一 F 被重复提交， C 能够返回存储在表 tbl 中的结果。 A 提交一个文件 $F = (m_1, \dots, m_n)$ ， C 运行 $AuthGen$ 算法输入 (sk_{cnt}, F) ，得到一个相应的验证者集 $\{D_i\}$ 和公共验证参数 ϕ_F ， C 向表 tbl 中添加一行 $(j, F, \{D_i\}, \phi_F)$ ，并将 j 的值加 1。

GenChallenge. 这个查询只能被发出一次，有时 A 提交两个同样长度的文件 F_0 和 F_1 （假设它们是区别于 $AuthQuery$ 的所有输入）， C 掷币（随机地选择） $b \in R\{0, 1\}$ ；并且运行 $AuthGen$ 算法，输入 (sk_{cnt}, F_b) 后，得到相应的认证 $\{D_i\}$ 和验证参数 ϕ_{F_b} ， C 设置这个挑战行 R 为 $(0, F_b, \{D_i\}, \phi_{F_b})$ ， ϕ_{F_b} 被返回给 A 。

ProofQuery(j). 在证明协议中 A 与 C 的博弈。注意到 j 被 A 指定并且指向表 tbl 中的第 j 行，或者与挑战行相关（ $j = 0$ 时，假设 $GenChallenge$ 查询已经发生了）。 A 扮演了 TPA 角色而 C 扮演了证明者角色，当 $j > 0$ ， C 输入为 $(pk_{cnt}, F, \{D_i\}, \phi_F)$ ，其中 $(j, F, \{D_i\}, \phi_F)$ 是表 tbl 中的一行，否则 $GenChallenge$ 查询已经生成，当 $j = 0$ ， C 输入 $(pk_{cnt}, F_b, \{D_i\}, \phi_{F_b})$ 。

Update. 当 A 调用这个查询时， C 首先设置 $cnt = cnt + 1$ ，接下来 C 再次调用 $KeyGen$ 算法得到一个新密钥对 (pk_{cnt}, sk_{cnt}) ，对于表 $tbl \cup R^*$ 中的每一行 $(F, \{D_i\}, \phi_F)$ ， C 调用 $AuthUpdate$ 算法 $(sk_{cnt-1}, pk_{cnt-1}, sk_{cnt}, pk_{cnt}, \{D_i\}, \phi_F)$ 以得到 $(\{D'_i\}, \phi'_F)$ ，把该行更新为 $(F, \{D'_i\}, \phi'_F)$ 。然后， C 向 A 返回 sk_{cnt-1} 。

Guess，此时 A 将输出一个猜测比特 b' 。如果 $b = b'$ ， A 将以 ϵ 的概率赢得这个游戏。其中 $|\epsilon - 0.5|$ 是可忽略的概率。如果对手 A 的多项式时间的优势可以忽略不计的话，云数据完整性验证策略就具备零知识数据隐私。

5.2.3 设计目标

为了在公开的云数据完整性验证方案中支持安全、高效的密钥更新和数据隐私保护，方案具有以下设计目标。

1. 功能性，在不影响云数据完整性检验的同时支持密钥更新。
2. 高效性，与一般的验证者更改方案相比较，该方案的通信和计算成本大幅降低。
3. 正确性，使用挑战信息、挑战数据块和有效公钥产生的所有有效证据使得证明协议输出为 1。
4. 可靠性，任意证明者能通过完整性验证方案使得 TPA 确信它所存储的文件，确实是该存储文件本身。
5. 零知识隐私，TPA 不会从公共信息，验证标签更新和与云服务器交互证据中得到存储文件的任何内容信息。

5.3 提出的方案

CrsGen(1^k). 输入一个安全参数 λ , 这个算法输出一个以大素数 p 为阶的两个同阶乘法循环群 G , G_T , 其中 g 是 G 的生成元。 $e : G \times G \rightarrow G_T$ 表示一个双线性映射, $H_0, H_1 : \{0, 1\}^* \rightarrow G$ 为两个单向哈希函数。另外, 算法随机选择 $h, u_1, u_2, \dots, u_s \in G$, 并且计算 $\eta = e(g, h)$ 。公共参数字符串 crs 是 $(k, p, G, G_T, g, e, H_0, H_1, h, u_1, u_2, \dots, u_s, \eta)$ 。

KeyGen(crs). 输入公共参数 crs , 数据所有者(云用户)生成一个签名密钥对 (spk, ssk) , $spk = g^{ssk}$ 和另外一个用于产生文件块认证的密钥对 (α, v) , 其中 $\alpha \in Z_p$, $v = g^\alpha$, 数据所有者的秘密密钥是 $sk = (\alpha, ssk)$, 公共密钥是 $pk = (spk, v)$ 。

为便于表达, 使用 η_i 代表 $e(u_i, v)_{1 \leq i \leq s}$ 。注意到通过相关的公钥 v 和 crs 能够预计算出: $\eta_1 = e(u_1, v) \cdots \eta_s = e(u_s, v)$ 。

AuthGen(sk, F). 给出一个文件 F , 数据所有者首先用例如 *RS* (*Reed – Solomon*) 的冗余编码处理文件 F 获得处理过的文件 F' , 并把文件 F' 分成 n 个块。每个块被进一步分成 s 个扇区 $\{m_{ij}\}_{1 \leq i \leq n, 1 \leq j \leq m}$, 扇区块 $m_{ij} \in Z_p$ 。数据所有者从一个足够大的域中选择一个名为 Fn 的文件。使得 t_0 为 $Fn||n$ 。数据所有者计算 $t = (H_0(t_0))^{ssk}$ 并将文件标签记为 $ft = t_0||t$ 。然后对于每一个块 i , $(1 \leq i \leq n)$ 数据所有者计算数据块 i 的验证标签 σ_i :

$$\sigma_i = (H(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^\alpha \quad (5-1)$$

最后, 数据所有者将: $ft||\{m_{ij}\}_{(1 \leq i \leq n, 1 \leq j \leq m)}||\{\sigma_i\}_{1 \leq i \leq n}$ 存储到云服务器。(需

注意的是，这里存在严格的访问控制策略，来判定访问存储文件和验证标签的用户是否合法。) 过程见图 5-2。

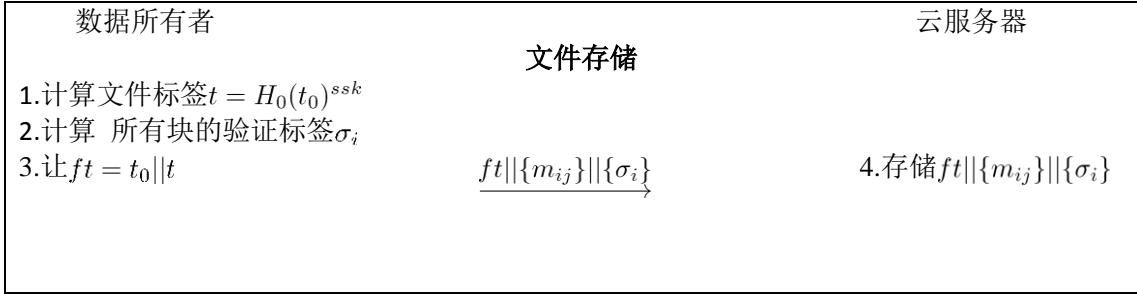


图 5-2 文件存储

$Proof((P(F, \{\sigma_i\}, ft), V(pk)))$ 由一个证明者(云服务商)与验证者 (TPA) 之间进行的 5 步交互证明协议执行生成，过程见图 5-3:

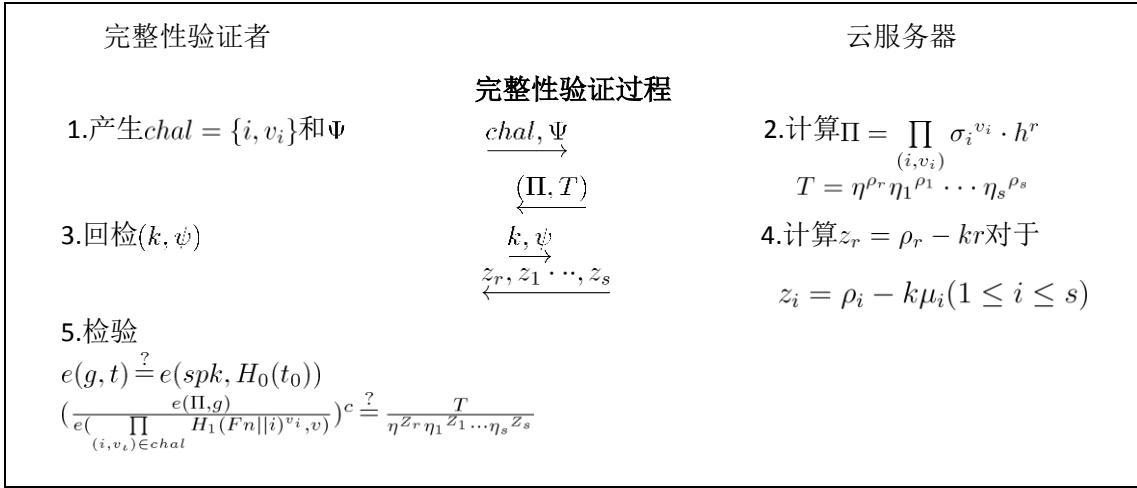


图 5-3 数据完整性验证过程

1. TPA 选择一个随机整数 c 和 $k, \psi \in Z_p$ ，计算 $\Psi = g^k h^\psi$ 。对于($1 \leq i \leq c$)TPA 选择一个随机的 $v_i \in Z_p$ 。然后把完整性验证过程中固定挑战数据块位置的 Ψ 和 $chal = \{i, v_i\}_{1 \leq i \leq c}$ 发送给云服务器。

2. 一旦收到($chal, \Psi$)后，云服务器首先随机地选择 $r, \rho_r, \rho_1, \dots, \rho_s \in Z_p$ ，再计算

$$\Pi = \prod_{(i, v_i) \in chal} \sigma_i^{v_i} \cdot h^r, T = \eta^{\rho_r} \eta_1^{\rho_1} \dots \eta_s^{\rho_s} \quad (5-2)$$

然后将 (T, Π) 发送给 TPA.

3. TPA 将 (k, ψ) 发送给云服务器。

4. 服务器核对等式 $\Psi = g^k h^\psi$ 是否成立，如果等式不成立，服务中断。否则计算 $z_r = \rho_r - kr, \mu_i = \prod_{(i, v_i) \in chal} v_i m_{ij}, z_i = \rho_i - k\mu_i (1 \leq i \leq s)$ ，并将 $(z_r, z_1 \dots, z_s)$ 发

送给 TPA。

5. TPA 首先通过以下等式验证文件标签 ft :

$$e(g, t) \stackrel{?}{=} e(spk, H_0(t_0)) \quad (5-3)$$

如果验证失败，通过发送 False 拒绝，否则 TPA 验证以下等式是否成立:

$$\left(\frac{e(\Pi, g)}{e\left(\prod_{(i, v_i) \in chal} H_1(Fn||i)^{v_i}, v \right)} \right)^k \stackrel{?}{=} \frac{T}{\eta^{Z_r} \eta_1^{Z_1} \dots \eta_s^{Z_s}} \quad (5-4)$$

$KeyUpdate(pk, sk)$. 数据所有者能够产生一个新的密钥以改变他的密钥对 (sk, pk) ，结果是数据所有者有了一个已更新的公钥 $pk = (spk', v')$ 和一个已更新的私钥 $sk = (\alpha', ssk')$ ，其中 $spk' = g^{ssk'}$, $v' = g^{\alpha'}$ 。

$AuthEvolve(pk, sk, pk', sk', ft, \sigma_i)$. 数据所有者从云服务器下载 $ft||\{\sigma_i\}_{1 \leq i \leq n}$ ，更新文件标签与数据块验证标签如下:

1. 计算 $t' = t \frac{ssk'}{ssk}$, 并使 $ft' = t_0||t'$ 。
2. 对于 $(1 \leq i \leq n)$, 计算 $\sigma'_i = \sigma_i^{\alpha'/\alpha}$ 。
3. 上传 $ft'||\{\sigma'_i\}_{1 \leq i \leq n}$ 至云服务器。

具体过程如图 5-4:

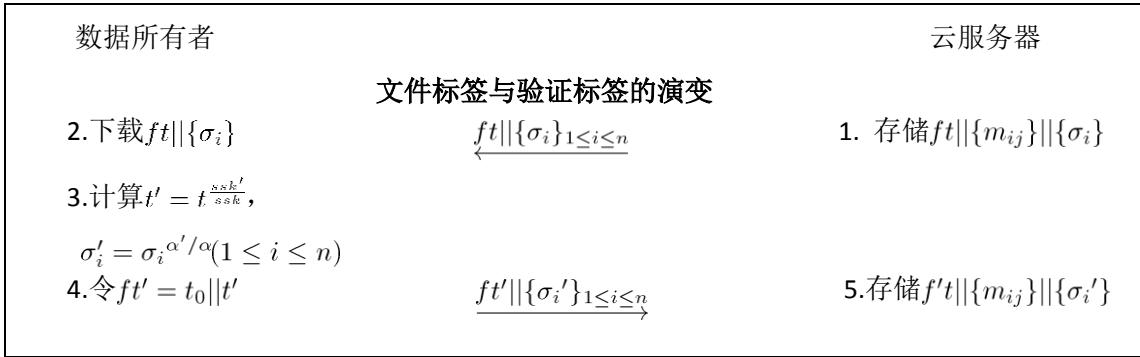


图 5-4 文件标签与验证标签的修改

5.4 安全性分析

此节展示了该方案实现的正确性，完整性和零知识隐私性。密钥更新和验证标签演变的有效性。完整性表明方案的正确性和在不可信服务器下的可确保安全性。

5.4.1 正确性

在密钥更新之前，方案的正确性是直接用双线性对来验证的。下面证明在密钥对改变之后验证仍然能够成立。如果数据所有者和服务器都是可信的，将得到:

$$\begin{aligned}
 t' &= t^{\frac{ssk'}{ssk}} \\
 &= ((H_0(t_0))^{ssk})^{\frac{ssk'}{ssk}} \\
 &= H_0(t_0))^{ssk'}
 \end{aligned} \tag{5-5}$$

因此， $e(g, t') = e(spk', H_0(t_0))$ 成立。

关于验证标签的演变，

$$\begin{aligned}
 \sigma'_i &= \sigma_i^{\alpha'/\alpha} \\
 &= ((H(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^\alpha)^{\alpha'/\alpha} \\
 &= (H_1(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^{\alpha'}
 \end{aligned} \tag{5-6}$$

$$e(\sigma', g) = e\left(\prod_{(i, v_i) \in chal} (H_1(Fn||i)^{v_i} \cdot \prod_{j=1}^s u_j^{\mu_j}, v')\right) \tag{5-7}$$

因此，上述公式 (5-7) 也成立。

5.4.2 可靠性

如果任意的伪造者都不能够欺骗验证者相信它所存储的文件 F 确实是该文件本身，那么这个完整性验证方案就具有可靠性。换句话说，对于伪造者，存在一个知识提取算法能够恢复文件块。方案的可靠性证明主要依赖于 Shacham-Waters 方案的可靠性。具体来说，证明了如果存在一个能够破坏本方案框架可靠性的对手，就能够构造出另一个算法来破坏 Shacham-Waters 方案的可靠性^[25]（参考 SW 方案提出）。

假设存在一个能够破坏本方案可靠性的对手 A ，就能构造一个能够破坏 SW 方案可靠性的模拟器 B 。

1. B 在 SW 方案中被给与一个公钥，其值 h 由 B 以 g 为底的离散对数的形式生成， g 是 SW 方案中公钥的一个已知元素， h 和 pk^* 中相应的元素将被看作 crs ，仅仅 SW 方案中公钥的 (v, spk) 被作为 pk^* 。令 A 更新查询的次数为 n ， B 选择一个随机索引 $\hat{i} \in \{1, \dots, n\}$ ，设 $pk_{\hat{i}} = pk^*$ 。对于 $i \in \{1, \dots, n\} \setminus \{\hat{i}\}$ ， B 选择一个随机 sk_i 并计算 $pk_i = g^{sk_i}$ 。 pk_i 将会作为这个系统的第一个公钥传送给 A 。

2. (除了周期 \hat{i} 的周期查询) 在不等于 \hat{i} 的周期，由于 B 拥有私钥，因此可以回答来自对手的所有查询。

3. (更新查询 $cnt = \hat{i} - 1$) 当 $cnt = \hat{i} - 1$ 时， A 调用更新查询， B 没有相应的私钥，因此要回答这个查询需要利用原始的 SW 方案的 $AuthQuery$ 算法以重新计算所有的验证标签。注意到，这些验证标签就好像是从更新查询步骤中被计算并分发一样。

4. (更新查询 $cnt = \hat{i}$)。当 $cnt = \hat{i}$ 时, A 调用更新查询, B 使用私钥 $sk_{\hat{i}+1}$ 重新计算所有的验证标签来回答这个查询。这些验证标签就好像是从更新查询步骤中被计算并分发一样。

5. (输出) 最后, A 输出一个证明 P_* 。 P_* 在周期 \hat{i} 以 $1/n$ 的概率输出, 否则 B 就中止。下面展示一下 B 是怎样向原始的 SW 方案输出一个证明 P' 。

6. (P' 的结构) P_* 被执行一次, B 得到 $(T, \Pi, k, z_r, z_1, \dots, z_s)$, B 将 P_* 返回第三步, 并用不同的值 k' 回应 (这是可能的, 因为 B 能够获得以 g 为底的离散对数 h , 并因此提供相应的 ψ 值)。现在这个方案以一个不同的记录 $(T, \Pi, k', z'_r, z'_1, \dots, z'_s)$ 被完成。从这两组等式中, B 能够获取基础值 (r, u_1, \dots, u_s) 。 B 计算 $\sigma = \Pi/h^r$ 并输出 $(\sigma, u_1, \dots, u_s)$ 代表证明 P' 。注意到如果 P_* 是 ϵ -admissible, 对于底层 SW 方案, P' 是 ϵ^2 -admissible, 换句话说, 如果 SW 方案可靠, 那么本章的方案也是可靠的。

5.5 性能分析与实验

在这一节中, 给出改进后方案的通信和计算的复杂度分析, 以及存储代价分析, 并随后给出实验结果。

5.5.1 参数选择

安全参数 λ 的典型选择 80, 由于这个方案是公开验证, p 应该是一个 $2k=160$ 比特的素数, 选择椭圆曲线使离散对数是 $2^k - \text{secure}$ 。将 λ 的值上升到 128, 归功于 Barreto 和 Naehrig^[116]配对性较强的素数阶椭圆曲线能够被运用。 n 表示一个文件中块的数量, 且远远大于 k 。遵循 SW 方案的建议来选择纠删码。传统的 RS 纠删码能够用于这个架构的实施, 但是加密和解密过程需要花费 $O(n^2)$ 时间。对于一个没有恶意的服务器, 为使系统代码可用于更有效的公共检索, 其中编码文件的最初的 m 个数据块就是文件本身。

5.5.2 复杂性分析

通信成本: 在 $proof$ 阶段, TPA 发送 ϕ 和 $chal$ 到云服务器, 它们的二进制比特长度为 $\log_2 c + (c + 1)\log_2 p$ 。通过选择一个伪随机置换去计算这些挑战块的位置 i 和一个伪随机函数来生成随机挑战值 v_i , 这样能大大地缩短这个挑战的长度。在这个前提下, TPA 只需要发送这个伪随机置换和伪随机函数的密钥, 它们每个的长度仅为 $\log_2 p$ 比特。在第二步, 云服务器向 TPA 返回椭圆曲线的两个点: T 、 Π , 它们的长度为 320 比特。在第三步, TPA 向云服务器发送 (k, ϕ) , 其二进制长度为 $2\log_2 p$ 。接下来, 云服务器向 TPA 发送 (z_r, z_1, \dots, z_s) , 其二进制长度为 $(s + 1)\log_2 p$ 。

存储成本：在该方案的存储成本方面，由于需要公共验证，文件和相应的数据元包括文件标签和块的验证标签都需要被存储在云服务器上。本章方案在存储和通信之间具有灵活的折中性^[25]。因此，参数 s 被用来在响应长度和存储开销之间获取一个折中的值。每一个块由 s 个属于 Z_p 的元素组成，这些元素被称为扇区。每一个块取代每一个扇区拥有有一个验证标签，这样就将存储开销减小到 $(1 + \frac{1}{s}) \times$ 。数据所有者只需要存储公钥 $pk = (spk, v)$ ，私钥 $sk = (\alpha, ssk)$ ，因此数据所有者的存储开销近似为 $2\log_2 p + 320$ 比特。TPA 需要存储一个用户的公共密钥，其二进制长度为 320 比特。在完整性验证过程中，TPA 需要存储 $k, \phi, chal, \Phi, T, z_r, z_1, \dots, z_s$ ，来确认一个来自服务器的响应是否有效，其总共二进制长度为 $(c + s + 3)\log_2 p + 320$ 比特。

计算成本：从数据所有者、云服务器、TPA 的角度给出计算成本。仅仅考虑一些昂贵的计算操作包括：双线性映射、 G 和 G_T 中的指数运算和乘法运算。 T_{pair} 表示计算一个椭圆曲线两个点的双线性映射的时间成本。同时忽略一些高效的操作，比如快速计算一个哈希函数的时间成本。 T_{exp_g} 、 $T_{exp_{gt}}$ 分别表示 G 和 G_T 中指数运算的时间成本。 T_{mul_g} 、 $T_{mul_{gt}}$ 分别表示 G 和 G_T 中乘法运算的时间成本。

数据所有者的主要计算是为文件块产生验证标签：
 $\sigma_i = (H_1(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^\alpha$ ，其时间开销为 $(s + 1)T_{exp_g} + sT_{mul_g}$ 。对于验证标签演变，数据所有者需要在 G 中执行 $n + 1$ 次指数运算，时间开销为 $(n + 1)T_{exp_g}$ 。据此推论，在方案中数据所有者的主要计算成本为 $(sn + 2n + 1)T_{exp_g} + nsT_{mul_g}$ 。为了产生和验证一个证明，TPA 需要计算 ϕ 和有效的文件标记和响应，因此 TPA 的整个计算成本为 $4T_{pair} + (c + 2)T_{exp_g} + (s + 2)T_{exp_{gt}}$ 。为了产生一个证明，云服务器不得不计算 Π, T 和 z_r, z_1, \dots, z_s ，云服务器的主要计算成本为 $(c + 2)T_{exp_g} + (c + 1)T_{mul_g} + (s + 1)T_{exp_{gt}} + sT_{mul_{gt}}$ 。

5.5.3 实验结果

实验平台如下：由pbc-0.5.13^[117]和pbc wrapper-0.8.0^[118]在Intel i7-4700MQ CPU @ 2.40GHz上执行该实验。由于这个方案仅需要一个多项式空间，因此内存始终是充足的。这个实现的源代码是可用的^[119]，在本节实验中，利用了参数a.param，它为pbc库的标准参数环境之一，这个参数提供了一个在所有默认参数中最快的对称对匹配。实验的协议开销在以下四个方面展显。

第一部分由该协议的离线阶段组成，包括常用公共参数字符串生成，密钥生成和认证标签生成。在这个实验中，假设一个文件的大小恒定为1MB。方案要求每一个扇区都是 Z_p 的一个群元素，其大小为160比特或20字节，整个扇区的数量约

为500000。在这个案例中仅有的变量是块的数量 n ，它也控制了每个块中扇区的数

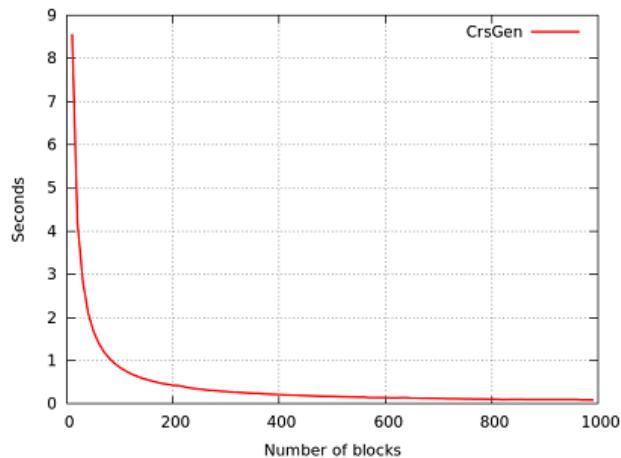


图 5-5 crs生成时间

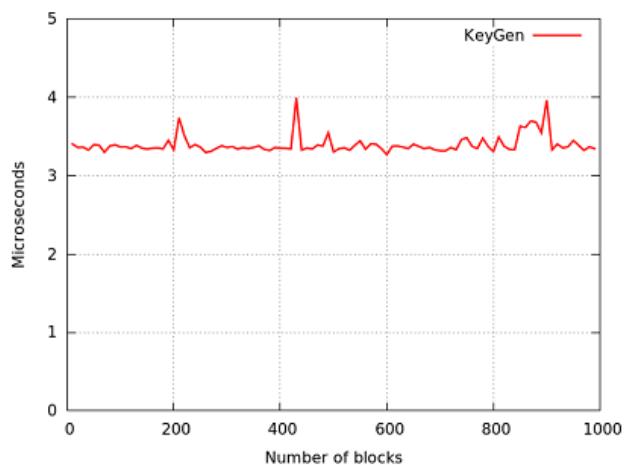


图 5-6 密钥生成时间

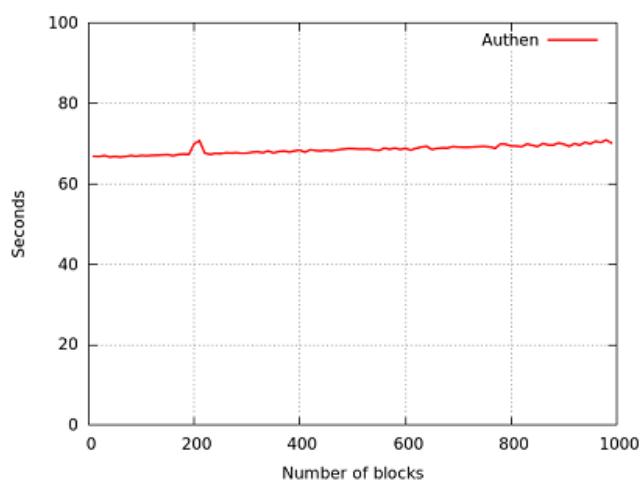


图 5-7 验证标签生成时间

量 s 。对于每一步，以 10 为增量，从 10 至 1000 线性地增加块的数量。注意到在密钥产生阶段，不应该包括计算 η_i 的时间，因为用户不需要自己去产生这些值，它包含在最开始系统计算生成的公共参数 crs 里。

结果如图 5-5, 5-6 和 5-7 中所示，可以看到生成 crs 的时间与每个块中扇区的数量 s 呈线性关系。其它计算无论块的数量为多少都是比较稳定的。这与经验分析是一致的，因为在生成 crs 的时候，主要的操作是生成 $u_1, u_2, \dots, u_s \in G$ ，其中 s 是一个块中扇区的数量。对于具有固定大小的文件，随着块数量的增加，每个块中的扇区数减少，这导致了更少的 u_i 生成。因此，生成 crs 的时间就减少了。在 *KeyGen* 算法中，仅仅涉及两个指数运算，这仅仅带来了常量的时间开销。开销最主要的部分应该是为一个文件产生验证标签。对于一个 1MB 的文件需要花费 70 秒来计算验证标签。这是合理的，因为对于一个固定大小的文件，总共扇区的数量也是固定的。结果，整个文件的主要计算 $\prod_{j=1}^s u_j^{m_{ij}}$ 的开销也是一个常数。但是可以肯定的是时间随着文件大小的增加几乎呈线性增加，因为扇区数和文件大小之间保持一个线性关系。

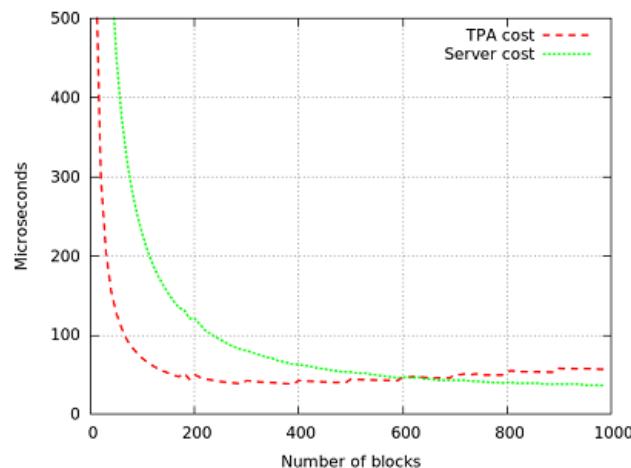


图 5-8 挑战 1% 的数据块

在实验的第二部分。对于一个给定挑战试图去确定块的数量。把文件的大小固定为 1MB，并且把挑战的数量(c)分别设为块数量的 1% 和 5%。挑战块的数量对于恰当的抽样检验方式检验损坏是足够的。Ateniese 等在文献[120]中给出了一个这样的例子，对于一个有 10000 个块的文件，如果文件中 1% 的块已经被损坏，那么，TPA 随机选择 460 个块就能以大于 99% 几率检测出服务器的错误。在这种情况下，挑战块只占整个文件的 4.6%。

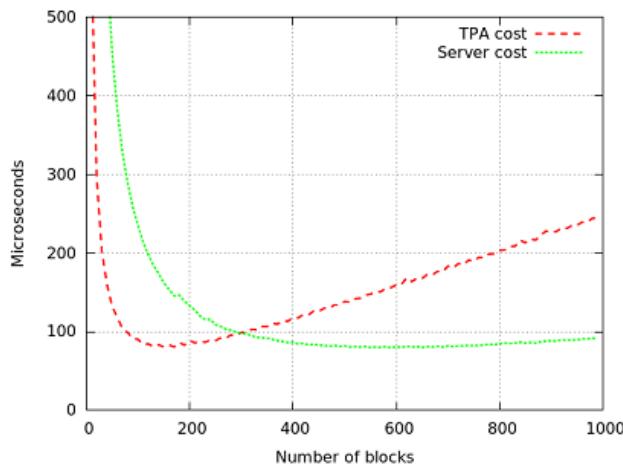


图5-9 挑战5%的数据块

表 5-1 算法的时间开销

参数			离线开销				在线开销	
s	n	c	crsGen	KeyGen	AuthenGen	TPA load	Server load	
83	600	1% n	144.0ms	3.3ms	68.8s	47ms	47.1ms	
166	300	5% n	284.0ms	3.4ms	67.6s	99.1ms	97.8ms	

在 $proof$ 阶段的第二步，服务器需要执行 $(s + c)$ 次指数运算。如图5-5,5-6,5-7中所示，随着块数量的增加，由于扇区数量(s)的减少，服务器的负担大大降低了，然后随着挑战块的数量(c)的增加，TPA负担又加大了。因此，应该选择一个服务器和TPA开销最优点。显然如图5-8，当挑战一个文件所有块的1%时，1MB大小的文件最佳块数量为620,这意味着每个块将近1.6KB。然而如图5-9，当5%的块被挑战时，文件的最佳块数量为300，这意味着每个块将近3.3KB。同时，当5%的块被挑战时，服务器的错误被发现的概率极大。因此，建议在实践中块的大小为3KB，这样可以服务和TPA的开销同时最小化。方案中的，每一个算法的时间开销总和总结如表5-1中。

确定最佳的块规模后，实验的第三部分测试证明协议的高效率性，这部分经常被用来检查云数据的完整性。使用具有10000个块的文件做测试，每个块的大小是3KB(150个扇区)。通过以10为增量，从10到1000增加随机选择的挑战块的数量，同时记录TAP和云服务器的时间代价。结果如图5-10中所示：证明协议对二者来说都是非常有效的。

比如，研究这个曲线上的两个点，Ateniese等人证明了，如果服务器有1%块被破坏，TPA可以挑战300个块和460个块达到对服务器异常行为至少分别有95%和99%的检测概率。可以看到，当挑战300块时，TPA花费1.42秒，服务器花费0.48秒；

而当挑战460块时，TPA花费2.16秒，服务器花费0.7秒。随着挑战块的增加，时间开销全面增加。这与经验分析相一致，当挑战块的数量上升时，TPA在计算 $\prod_{(i,v_i) \in chal} (H_1^i(Fn||i))^{v_i}$ 时会有更多的指数运算和乘法运算操作，云服务器需要计算更多的 $\sigma_i^{v_i}$ 。

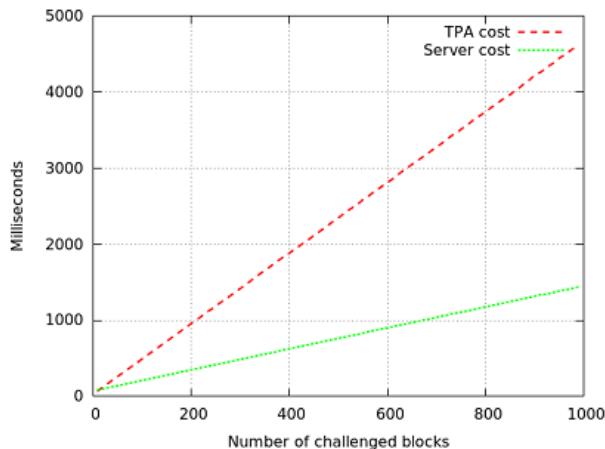


图5-10 将文件的挑战数据块一直增加到10000

表 5-2 通信开销与验证标签更新时间开销比较

	原始的解决方案	我们的解决方案
通信开销	1 MB	<20KB
验证标签更新时间开销	约70s	<1s

在实验的最后部分，比较本文方案与原始的密钥更新方案，即数据所有者从云端下载整个文件并重新生成验证标签。这意味着用户不得不重复完成AuthenGen算法过程。另一方面，在协议中，数据所有者需要上传 n 个验证标签，它们是 G 中的群元素。在实验中，我们选择 $n=300$ ，而对a.param中的每一个元素都存储为64字节。因此，使用我们的方案通信开销要少于20KB，在验证标签更新时间开销方面，使用原始模型需要重新计算所有的验证标签，这将花费70秒左右的时间，而使用我们的模型如实验的第一部份所示仅需要完成约 n 次指数运算。对于 $n=300$ ，它需要不到1秒的时间，结果见表5-2。显而易见，在计算和通信方面，我们协议与原始方案相比更具有效率优势。

5.6 本章小结

在本章中，为了让云存储更加实用，研究了关于云存储安全完整性验证的两个重要问题：（1）当一个云用户的密钥改变时如何有效地改变外存文件的验证标签。（2）如何在密钥更新完整性验证方案中保护存储文件的隐私。通过形式化支

持密钥更新的云存储数据完整性验证过程的可靠性与零知识数据隐私的安全模型，并结合 Shacham-Waters 方案和一些新的密码技术，提供了一个可靠的框架结构。在新的安全模型中，对该方案的可靠性和零知识隐私性进行证明。性能分析和实验结果表明，新方案是高效实用的。

第六章 密文类型可修改的云存储数据可用性方案

本章在前面三章对云存储数据完整性研究外开展对云存储数据的可用性研究。在实际的云存储应用中，密文类型信息的动态可修改性具有广泛的应用场景。最近针对这种应用需求 Liu 等人^[58]在 Ibraimi 等人^[53]提出的基于类型和身份的代理重加密方案基础上，提出了一种类型可修改的基于身份代理重加密方案。该方案在具有传统代理重加密方案的核心功能的同时，还具有密文的拥有者随时修改密文类型信息的功能。但是经过仔细分析发现该方案存在两个安全漏洞：1、类型修改缺乏验证，攻击者可以随意修改类型标记；2、类型修改引起了新的条件性选择明文攻击问题。在本章中将在分析这两个安全漏洞的基础上提出改进方案并给出安全性分析。

本章内容由四部分构成：首先，6.1 节简要介绍相关研究与系统模型；其次，6.2 节简要介绍 Luan Ibraimi 等人的基于身份代理重加密方案与 Liu 等人提出的一种类型可修改的基于身份代理重加密方案；再次，6.3 节对 Liu 等人方案存在的安全漏洞进行分析，并在 Liu 等人方案基础上进行了进一步的改进，提出了一个对类型可修改的基于身份代理重加密方案的改进方案，并对方案的安全性进行了分析；最后，6.4 节对本章内容进行了小结。

6.1 相关研究与系统模型

6.1.1 相关研究

云存储中，代理重加密（Proxy Re-Encryption, PRE）^[1]技术可以保障用户数据在存储第三方的安全性和可共享性。该技术的核心思想是：数据拥有者以密文形式将数据存储在第三方；数据拥有者可以委托存储第三方对其存储的密文进行重加密并共享给其他用户。

最近 Liu 等人针对现实应用在 Ibraimi 等人^[52-53]提出的基于类型和身份的代理重加密方案基础上提出一种类型可修改的基于身份代理重加密方案^[54-57]。此方案不但实现了 PRE 中细粒度的密文共享，还解决了原方案密文类型信息静态，不能动态修改问题。Liu 等人敏锐的发现了密文类型信息动态修改的现实意义与应用场景。比如：数据拥有者为节省本地存储开销，或方便数据在不同终端的灵活使用，将标记为“不可共享”类型的加密数据存储于服务器。数据拥有者根据现实需要希望将“不可共享”类型修改为“可共享”类型；一段时间后又需要将“可共享”

类型恢复为“可共享”。类型信息的动态修改就可以很好的适应此类应用。

Liu 等人的方案^[58]虽然具有很高的现实应用价值，并且根据其安全性分析声称具有与文献[53]方案相同的安全性。但是在对其所提方案进行仔细研究后发现如图 6-1 所示在完成密文类型信息的修改后并没有对密文类型信息进行验证。其方案存在两个安全漏洞：1、类型修改缺乏验证，攻击者可以随意修改类型标记，而不会被接收方发现。2、由于类型修改引起了新的条件性选择明文攻击问题。

6.1.2 系统模型

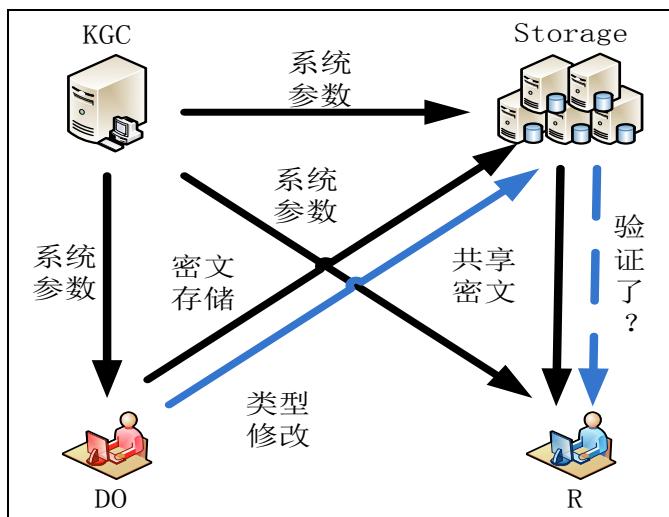


图 6-1 系统模型

该模型包括 4 个主体：

1. 数据拥有者 (Data Owner, DO): 数据所有者本身，具体而言就是具有存储服务器存储数据所有权限的用户。
2. 数据共享者 (Receiver, R): 被数据拥有者授权可以对数据拥有者存储在服务器上的共享数据进行访问的用户。
3. 密钥生成中心 (KGC): 用于生成方案中所需的相关公开参数和秘密参数，并将其发送与各实体。
4. 第三方存储器 (Storage): 将所有的相关分布式云存储器视为具有一个管理者进行管理的实体存储设备，用来存储数据拥有者的数据。因为数据拥有者并不需要关心这个实体存储设备到底是如何运行的，所以在研究模型中将之视为一个整体。

模型应用主要包括四个步骤：

1. 系统参数设置 (*Setup*): 在系统参数设置过程中，KGC 生成 PRE 的系统参数，并将之分配与各个实体。

2. 数据加密存储 (*Storage*): 在数据存储过程中, DO 先对数据进行处理, 然后将加密后的密文信息存储在 Storage 上。

3. 密文共享 (*Shared*): 在数据共享过程中, 首先 DO 根据数据共享者 R 的公钥信息生成重加密密钥, 并将其发送给 Storage; 然后 Storage 根据该密钥对预共享的密文进行重加密, 并将其生成的重加密密文发送给 R, 从而实现密文的共享。在数据的存储与共享过程中, PRE 保证即使 Storage 是不可信的, Storage 仍然不可能获取数据的任何内容信息。

4. 类型修改 (*Update*): 在密文类型修改过程中。DO 将新的密文类型与由新的类型生成的密文类型修改密钥传给 Storage。Storage 收到密文类型修改密钥后对数据加密存储阶段存储于 Storage 的密文进行处理。处理生成具有新密文类型的预共享密文, 然后经过密文共享阶段的重加密, 最后实现密文类型修改后的密文共享。

6.2 类型可修改的基于身份代理重加密方案

6.2.1 对 Luan Ibraimi 等人方案的回顾

本节将简单回顾由 Luan Ibraimi 等人[53]提出的基于类型和身份的代理重加密方案。该方案由以下七个算法组成:

算法 1. $Setup(1^k)$: $PK = (G, G_T, g, Pub = g^\alpha, \hat{e}, H_1, H_2)$ 是由 KGC 生成主公开参数, 其中 G 和 G_T 为大素数 p 阶群, g 为生成元, $\hat{e} : G \times G \rightarrow G_T$; 主秘密参数 $MK = \alpha$, $\alpha \in Z_p^*$, hash 函数 $H_1 : \{0, 1\}^* \rightarrow G$ 和 $H_2 : \{0, 1\}^* \rightarrow Z_p^*$ 。

算法 2. $Extract(MK, ID)$: KGC 生成身份信息 ID 所对应的私钥 $SK_{ID} = H_1(ID)^\alpha$ 。

算法 3. $Enc(PK, ID, SK_{ID}, t, m)$: 该算法由明文 m 的拥有者采用自己的身份信息 ID 生成密文 $c = (c_1, c_2, c_3)$: (1) 随机选取 $r \in Z_p^*$, $c_1 = g^r$; (2) $c_2 = m \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t)}$; (3) $c_3 = t$ 。

算法 4. $Dec(c, SK_{ID})$: 该算法由私钥的拥有者对密文 c 进行解密:

$$m = \frac{c_2}{\hat{e}(SK_{ID}, c_1)^{H_2(SK_{ID}, c_3)}}.$$

算法 5. $RKey(ID_i, ID_j, t, SK_{ID_i})$: 由 DO 执行:(1) 随机选择 $X \in G$ 和 $r' \in Z_p^*$ (2) 计算: $RK_{ID_i \rightarrow ID_j} = (t, SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot H_1(X), g^{r'}, X \cdot \hat{e}(H_1(ID_j), Pub)^{r'})$ 。

算法 6. $REnc(c, RK_{ID_i \rightarrow ID_j})$: Storage 执行, 将预共享密文 c 转变为可由共享者 ID_j 解密的密文 $c' = (c'_1, c'_2, c'_3, c'_4)$, 具体的执行过程如下: (1) $c'_1 = c_1$; (2) $c'_2 = c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot H_1(X)) = m \cdot \hat{e}(g^r, H_1(X))$; (3) $c'_3 = g^{r'}$; (4)

$$c'_4 = X \cdot \hat{e}(H_1(ID_j), Pub)^{r'}.$$

算法 7. $RDec(c', SK_{ID_j})$: 由 $R(ID_j)$ 解出 m : (1) $X = \frac{c'_4}{\hat{e}(c'_3, SK_{ID_j})}$; (2) $m = \frac{c'_2}{\hat{e}(c'_1, H_1(X))}$ 。

6.2.2 类型可修改的基于身份代理重加密方案

通过上述方案描述可知, 文献[53]提出的方案是不能实现密文类型的动态修改的。当密文 c 的数据拥有者需要对密文类型进行修改时, 该算法没有给出具体解决办法。而 Liu 等人在经典的代理重加密方案^[52-58]中添了一个新的功能: 密文类型的修改功能。在密文类型修改阶段 DO 将新的密文类型与由新的类型生成的密文类型修改密钥传给 Storage。Storage 收到密文类型修改密钥后对数据加密存储阶段存储于 Storage 的密文进行处理。处理生成具有新密文类型的预共享密文, 然后经过密文共享阶段的重加密, 最后实现密文类型修改后的密文共享。

Liu 等人方案在 Luan Ibraimi 等人方案的基础上添加了 2 个算法组成:

算法 8. $TKey(t, t', SK_{ID_i})$: 由 DO 执行, 生成修改密文类型的密钥:
 $TK = (t', SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t')})$ 。

算法 9. $TSet(c, TK)$: 由 Storage 执行, 修改密文 $c = (c_1, c_2, c_3)$ 的类型, 并生成具有新类型的密文 $c' = (c'_1, c'_2, c'_3)$: (1) $c'_1 = c_1$; (2) $c'_3 = t'$; (3) $c'_2 = c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t')}) = m \cdot \hat{e}(c_1, SK_{ID_i}^{H_2(SK_{ID_i}, t')})$ 。

6.3 对类型可修改的基于身份代理重加密方案的改进

6.3.1 Liu 等人提出方案的安全性分析

Liu 等人的方案虽然能够实现云服务器上密文类型的动态更新, 但是该方案在解决问题的同时, 又引入了两个新的安全性漏洞:

第一个安全性漏洞: 数据类型修改缺乏一个认证过程, 导致攻击者能将数据类型任意修改, 比如将不可共享类型修改为可共享类型。

第二个安全性漏洞: 由于类型修改引起了新的条件性选择明文攻击问题。

6.3.1.1 类型修改缺乏验证

Liu 等人的方案^[58]在文献[53]的基础上添加了密文类型的修改功能。在密文类型修改阶段中 DO 可以生成用于修改密文类型的密钥, 并将其发送给 Storage; 根据接收到的密钥, Storage 可以修改密文类型的信息, 并生成新的预共享密文。在经过密文共享阶段的重加密后再生成共享密文, 最终实现数据的安全共享。仔细

分析整个工作流程，在添加了密文类型修改阶段后，方案直接使用了方案原来的密文共享步骤，并没有在后面重加密和解密步骤对修改后的密文类型的状态做出相应的验证。整个方案的完整性遭到了破坏。

由于类型修改缺乏验证，攻击者可以随意修改密文类型标记 t' 。将修改后 t' 代入后面的代理重加密过程和解密过程。如下面推导过程所示 t' 是否被修改与这两个过程是否正确执行并不相关。相应的服务器、数据持有者、数据共享对象也无法及时地发现密文类型是否被攻击者所修改。

1. 攻击者截获算法 8 中， $TK = (t', SK_{ID_i}^{-H_2(SK_{ID_i}, t')} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t')})$ 这个由数据拥有者执行算法 $TKey(t, t', SK_{ID_i})$ 生成的并传递于云存储服务器的修改密文类型密钥，并将其中的密文类型部分 t' 修改成 t'' 后传于云存储服务器。

2. 云存储服务器通过执行算法 9 中 $TSet(c, TK)$: $c'_1 = c_1$; $c'_3 = t''$, $c'_2 = c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t')} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t')}) = m \cdot \hat{e}(c_1, SK_{ID_i}^{H_2(SK_{ID_i}, t')})$ 。

3. 云存储服务器在收到具有新的密文类型的密文 $c' = (c'_1, c'_2, c'_3)$ 。其数据的代理重加密过程和解密过程与正常情况一致。由数据拥有者执行算法中 $RKey()$: 计算 $RK_{ID_i \rightarrow ID_j} = (t', SK_{ID_i}^{-H_2(SK_{ID_i}, t')} \cdot H_1(X), g^{r'}, X \cdot \hat{e}(H_1(ID_j), Pub)^{r'})$ 。攻击者截获上式内容后同样将 t' 修改成 t'' ，其余部分内容不变，然后上传于云存储服务器 $RK^*_{ID_i \rightarrow ID_j} = (t'', SK_{ID_i}^{-H_2(SK_{ID_i}, t')} \cdot H_1(X), g^{r'}, X \cdot \hat{e}(H_1(ID_j), Pub)^{r'})$ 。

4. 云存储服务器执行算法 6 中 $REnc(c, RK_{ID_i \rightarrow ID_j})$ 。得到 $c' = (c'_1, c'_2, c'_3, c'_4)$: $c'_1 = c_1$; $c'_2 = c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t')} \cdot H_1(X)) = m \cdot \hat{e}(g^r, H_1(X))$; $c'_3 = g^{r'}$; $c'_4 = X \cdot \hat{e}(H_1(ID_j), Pub)^{r'}$ 。

5. R 执行算法 7 中 $RDec(c', SK_{ID_j})$ 。解密得到明文 (1) $X = \frac{c'_4}{\hat{e}(c'_3, SK_{ID_j})}$; (2) $m = \frac{c'_2}{\hat{e}(c'_1, H_1(X))}$ 。

根据上面的推导，可以清晰的看到由于重加密过程与解密过程不涉及密文类型标识 t' ，因此可以对其进行任意修改。攻击者只需要针对密文类型修改密钥和代理重加密密钥的传输过程做出相应的篡改，就可以将原本的可共享类型修改为不可共享类型，将不可共享类型修改为可共享类型。

6.3.1.2 由于类型修改引起了新的条件性选择明文攻击问题

在 Liu 等人的文章[58]中提到其采用的 PRE 方案“即使 Storage 不可信，Storage 也无法知道数据的内容”。但是 Liu 等人的方案[58]在提供数据类型可修改功能的过程中实际上泄露了部分的数据信息内容，造成了新的选择明文攻击问题。

攻击者与服务器合谋，攻击者在 Storage 处获取到在密文类型为 t_1 情况下的密文： $c_{m_1} = (c_{1,m_1}, c_{2,m_1}, c_{3,m_1})$ 其攻击目标为恢复相应的明文 m_1 。下面攻击者与云服

务器合谋攻击步骤如下：

首先攻击者采用选择明文攻击，获取到密文类型为 t_2 情况下的明密文对 (m_2, c_{m_2}) : $c_{m_2} = (c_{1,m_2}, c_{2,m_2}, c_{3,m_2})$ 。

$$\text{其中 } c_{2,m_2} = m_2 \cdot \hat{e}(H_1(ID_i), Pub)^{r_2 \cdot H_2(SK_{ID_i}, t_2)} \quad (6-1)$$

根据前面得到的密文 $c_{m_1} = (c_{1,m_1}, c_{2,m_1}, c_{3,m_1})$

$$\text{其中 } c_{2,m_1} = m_1 \cdot \hat{e}(H_1(ID_i), Pub)^{r_1 \cdot H_2(SK_{ID_i}, t_1)} \quad (6-2)$$

$$\text{则有: } c_{2,m_1} \cdot c_{2,m_2} = m_1 \cdot m_2 \cdot \hat{e}(H_1(ID_i), Pub)^{r_1 \cdot H_2(SK_{ID_i}, t_1) + r_2 \cdot H_2(SK_{ID_i}, t_2)} \quad (6-3)$$

从上面可以看出如果密文类型不可修改，该方案是可以抵御选择明文攻击的，但是由于数据类型明文存放、传输并可修改，带来了新的问题。如果云服务器出于好奇，将历史的代理重加密密钥与密文类型转换密钥记录，其中密文类型转换记录如表 6-1 所示：

表 6-1 密文类型转换记录表

t_1	t_2	\dots	t_n
t_1	$t_1 \rightarrow t_1$		$t_1 \rightarrow t_n$
t_2		$t_2 \rightarrow t_2$	$t_2 \rightarrow t_n$
\vdots		\ddots	
t_n			$t_n \rightarrow t_n$

在表中除自身的转换外，每一项都对应了一次密文类型转换。这些项代表 Storage 记录了相应的密文类型转换所需的密文类型转换密钥。分析算法 8 中密文类型转换密钥 $TK = (t', SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t')})$ 得知，其数据结构只与 (SK_{ID_i}, t', t) 相关。这说明 Storage 对密文类型转换的操作是可以多次叠加的。将 $TK = ()$ 中的密文类型转换密钥 $SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t')}$ 记做 $RTK_{t \rightarrow t'}$ 。如果密文类型转换记录表中存在一个 t_1 与 t_2 共同指向的密文类型 t^* ，将由 t_1 转换到 t^* 中间所经历的密文类型记录为一个非空集合 $\{t_{\theta_1}, t_{\theta_2}, \dots, t_{\theta_l}\}$, $1 \leq l \leq n$ 。Storage 可以通过查询记录表获取所需的密文类型修改密钥组，这个密文类型修改密钥组表示为 $\{RTK_{t_1 \rightarrow t_{\theta_1}}, RTK_{t_{\theta_1} \rightarrow t_{\theta_2}}, \dots, RTK_{t_{\theta_l} \rightarrow t^*}\}$ 。对密文类型修改密钥进行乘积处理：

$$\begin{aligned}
 & RTK_{t_1 \rightarrow t_{\theta_1}} \cdot RTK_{t_{\theta_1} \rightarrow t_{\theta_2}} \cdots RTK_{t_{\theta_l} \rightarrow t^*} \\
 &= SK_{ID_i}^{-H_2(SK_{ID_i}, t_1)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t_{\theta_1})} \cdot SK_{ID_i}^{-H_2(SK_{ID_i}, t_{\theta_1})} \\
 &\quad SK_{ID_i}^{H_2(SK_{ID_i}, t_{\theta_2})} \cdots SK_{ID_i}^{-H_2(SK_{ID_i}, t_{\theta_l})} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t^*)} \\
 &= SK_{ID_i}^{-H_2(SK_{ID_i}, t_1)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t^*)} \\
 &= RTK_{t_1 \rightarrow t^*}.
 \end{aligned} \quad (6-4)$$

Storage 最终可以通过计算得到由类型 t_1 到共同类型 t^* 的密文类型转换密钥 $RTK_{t_1 \rightarrow t^*}$ 。再经过算法 9 进行密文类型转换：

$$\begin{aligned} c'^*_{2,m_1} &= c_{2,m_1} \cdot \hat{e}(c_{1,m_1}, SK_{ID_i}^{-H_2(SK_{ID_i}, t_1)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t^*)}) \\ &= m_1 \cdot \hat{e}(H_1(ID_i), Pub)^{r_1 \cdot H_2(SK_{ID_i}, t_1)} \cdot \\ &\quad \hat{e}(c_{1,m_1}, SK_{ID_i}^{-H_2(SK_{ID_i}, t_1)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t^*)}) \\ &= m_1 \cdot \hat{e}(H_1(ID_i), Pub)^{r_1 H_2(SK_{ID_i}, t^*)} \end{aligned} \quad (6-5)$$

得到新类型 t^* 下的预共享密文 c'^*_{2,m_1} ，同理可得由 t_2 转换到 t^* 情况下的预共享密文 c'^*_{2,m_2} ： $c'^*_{2,m_2} = m_2 \cdot \hat{e}(H_1(ID_i), Pub)^{r_2 H_2(SK_{ID_i}, t^*)}$ 。

接下来在代理重加密过程中服务器根据记录的代理重加密密钥：

$RK^*_{ID_i \rightarrow ID_j} = (t^*, SK_{ID_i}^{-H_2(SK_{ID_i}, t^*)} \cdot H_1(X^*), g^{r^*}, X^* \cdot \hat{e}(H_1(ID_j), Pub)^{r^*})$ 修改：

$$c'_1 = c'_{1,m_1} \cdot c'_{1,m_2} = g^{r_1} \cdot g^{r_2} = g^{r_1+r_2}; \quad c'_3 = c'_3 = g^{r^*}; \quad \text{令 } c'_2 = c'^*_{2,m_1} \cdot c'^*_{2,m_2}$$

攻击者最终将得到篡改过的共享密文 $c'^* = (c'_1, c'_2, c'_3, c'_4)$ 其中：

$$\begin{aligned} c'_2 &= c'_2 \cdot \hat{e}(c'_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t^*)} \cdot H_1(X^*)) \\ &= m_1 \cdot m_2 \cdot \hat{e}(H_1(ID_i), Pub)^{(r_1+r_2) \cdot H_2(SK_{ID_i}, t^*)} \\ &\quad \cdot \hat{e}(g^{r_1+r_2}, SK_{ID_i}^{-H_2(SK_{ID_i}, t^*)} \cdot H_1(X^*)) \\ &= m_1 \cdot m_2 \cdot \hat{e}(g^{(r_1+r_2)}, H_1(X^*)) \end{aligned} \quad (6-6)$$

$$c'_4 = X^* \cdot \hat{e}(H_1(ID_j), Pub)^{r^*}$$

攻击者最终通过合法解密过程算法 7 得到

$$\begin{aligned} X^* &= \frac{c'_4}{\hat{e}(c'_3, SK_{ID_j})} \\ m &= \frac{c'_2}{\hat{e}(c'_1, H_1(X^*))} = \frac{c'_2}{\hat{e}(g^{(r_1+r_2)}, H_1(X^*))} = m_1 \cdot m_2 \end{aligned} \quad (6-7)$$

由 m 可得 $m_1 = m/m_2$ 。

攻击者攻击成功，由上可见在一定条件下，即攻击者与云服务器合谋时，密文类型修改会引起新的选择明文攻击问题。

6.3.2 方案改进

针对上述两个安全漏洞，本节对 Liu 等人的方案做出了以下改进：1、增加一个对类型 t 的完整性验证环节；2、对数据块进行编号。

在密文存储阶段引入对数据块进行区分的编号信息 k 、 $(m_1, \dots, m_k, \dots, m_n \in G, 1 \leq k \leq n)$ ，并将算法 3 修改为 $Enc(PK, ID, SK_{ID}, t, m_k, k)$ ，其中 c_2 由 $c_2 = m \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t)}$ 改为 $c_2 = m_k \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t, k)}$ 。相应的：

算法 4 修改 $m_k = \frac{c_2}{\hat{e}(SK_{ID}, c_1)^{H_2(SK_{ID}, c_3, k)}}$ 。

算法 5 修改 $RKey(ID_i, ID_j, t, k, SK_{ID_i})$: 其中

$$RK_{ID_i \rightarrow ID_j} = (t, SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)} \cdot H_1(X)^t, g^{r'}, X \cdot \hat{e}(H_1(ID_j), Pub)^{r'})。$$

算法 6 中 $Storage$ 将传输 $c' = (c'_1, c'_2, c'_3, c'_4, c'_5)$ 其中:

$$c'_2 = c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)} \cdot H_1(X)^t) = m_k \cdot \hat{e}(g^r, H_1(X))^t; c'_5 = t。$$

算法 7 中 $X = \frac{c'_4}{\hat{e}(c'_3, SK_{ID_j})}$; $m_k = \frac{c'_2}{\hat{e}(c'_1, H_1(X))^{c'_5}}$ 。

算法 8 中 $TKey(t, t', SK_{ID_i})$:

$$TK = (t', SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t', k)})。$$

算法 9 中 $c'_2 = c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)} \cdot SK_{ID_i}^{H_2(SK_{ID_i}, t', k)})$

$$= m_k \cdot \hat{e}(c_1, SK_{ID_i}^{H_2(SK_{ID_i}, t', k)})。$$

6.3.3 正确性验证

通过上述改进的算法 4 与算法 7, 可以看到 R 可正常解密共享密文最终取得

m_k 。

1. Storage 存储: $c_2 = m_k \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t, k)}$

2. DO 解密正确性:

$$m_k = \frac{c_2}{\hat{e}(SK_{ID}, c_1)^{H_2(SK_{ID}, c_3, k)}} = m_k \cdot \frac{\hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t, k)}}{\hat{e}(SK_{ID}, g^r)^{H_2(SK_{ID}, t, k)}} = m_k \quad (6-8)$$

3. Storage 存储预共享密文通过代理重加密后转换成最终共享密文: $c_2 \rightarrow c'_2$

$$\begin{aligned} c'_2 &= c_2 \cdot \hat{e}(c_1, SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)} \cdot H_1(X)^t) \\ &= m_k \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t, k)} \\ &\quad \cdot \hat{e}(g^r, SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)}) \cdot \hat{e}(g^r, H_1(X))^t \\ &= m_k \cdot \hat{e}(g^r, H_1(X))^t; \end{aligned} \quad (6-9)$$

4. R 进行最后的数据解密:

$$X = \frac{c'_4}{\hat{e}(c'_3, SK_{ID_j})} = \frac{X \cdot \hat{e}(H_1(ID_j), Pub)^{r'}}{\hat{e}(g^{r'}, SK_{ID_j})} = X \quad (6-10)$$

$$m_k = m_k \cdot \frac{\hat{e}(g^r, H_1(X))^t}{\hat{e}(c'_1, H_1(X))^{c'_5}} = m_k \cdot \frac{\hat{e}(g^r, H_1(X))^t}{\hat{e}(g^r, H_1(X))^{c'_5}} = m_k \quad (6-11)$$

6.3.4 安全性分析

Liu 等人方案[58]安全漏洞的产生是由于只添加了密文类型修改功能, 而没有在其后的环节做出相应的验证, 使得方案整体的完整性遭到了破坏。针对密文类型修改所引起的两个安全漏洞本文在其方案相应的位置做出了修改, 下面通过对比来进行说明此修改是否会引起新的问题。

首先在密文存储阶段引入了对数据块进行区分的编号信息 k ，并将 c_2 由 $c_2 = m \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t)}$ 改为 $c_2 = m_k \cdot \hat{e}(H_1(ID), Pub)^{r \cdot H_2(SK_{ID}, t, k)}$ 。这样就对数据的密文与数据的编号信息进行了位置绑定，阻止了可能的指数项合并。在遭遇攻击时云服务器无法将不同编号数据块所对应的预共享密文顺利的改变为相同的类型结构，从而抵御了选择明文攻击。

其次为解决密文类型修改的验证问题，本文在代理重加密阶段将 $RKey()$ 中的 $RK_{ID_i \rightarrow ID_j}$ 由 $(t, SK_{ID_i}^{-H_2(SK_{ID_i}, t)} \cdot H_1(X), g^{r'}, X \cdot \hat{e}(H_1(ID_j), Pub)^{r'})$ 改变为 $(t, SK_{ID_i}^{-H_2(SK_{ID_i}, t, k)} \cdot H_1(X)^t, g^{r'}, X \cdot \hat{e}(H_1(ID_j), Pub)^{r'})$ 。

显然由于 $RKey()$ 是由 DO 进行运算的，在 $H_1(X)$ 的指数部分增加 t 并不会损害方案的安全性，但是却可以在算法 7 部分解密阶段与 $c'_5 = t$ 进行验证，如果 c'_5 被修改则 R 无法正确解密获取数据信息。

最后，根据文献[53]的安全性证明，与文献[58]的安全性分析。新方案对比旧方案只增加了密文与数据块明文编号信息的绑定，以及最后一阶段对明文传输密文类型信息 t 的验证，因此新方案具有与文献[53]一致的安全性。

6.4 本章小结

本章主要研究类型可修改的基于身份代理重加密方案。首先概述了类型可修改的基于身份代理重加密方案的相关研究与系统模型，指出在类型可修改流程结束后还需要对修改类型进行验证。接着，简要介绍了 Luan Ibraimi 等人的基于身份代理重加密方案与 Liu 等人提出的一种类型可修改的基于身份代理重加密方案。在对 Liu 等人方案进行安全性分析后指出：其方案存在两个安全漏洞：1、类型修改缺乏验证，攻击者可以随意修改类型标记；2、类型修改引起了新的条件性选择明文攻击问题。然后，针对上述安全漏洞在 Liu 等人方案基础上作出了进一步的改进，提出了一个类型可修改的基于身份代理重加密方案的改进方案。安全性分析表明新方案与传统的基于类型和身份的代理重加密方案具有相同的安全性。

第七章 全文总结与展望

7.1 全文研究工作总结

近年来，随着云存储应用的飞速发展，针对云端数据的完整性与可用性研究已成为当前云计算安全的热门研究课题。现有的云端数据完整性验证方案往往只关注于云端数据完整性验证过程中的可公开验证、数据隐私保护、数据动态操作、批量任务处理等一般性需求，而忽略了实际应用中对验证效率、用户可撤销、密钥更新、密文类型修改等可用性要求。因此亟需设计高效的云数据完整性验证方案，并以此为基础扩展可用性方案来提供相关应用场景的安全技术保障。本论文开展云数据完整性与可用性研究，并取得如下研究成果：

1. 高效的云存储数据完整性研究

设计了一个轻量级的云存储数据自我完整性验证方案。该方案中验证标签是基于椭圆曲线数据签名算法（ECDSA）构造的，因此验证过程不含双线性对，从而显著的减少了通信代价和计算开销。同时，方案通过一个哈希索引列表有效地支持数据的动态操作。

设计了一个高效的云存储数据公开完整性验证方案。该方案利用变型的 Schnorr 签名算法与同态消息认证码技术减少存储认证信息的存储空间与审计证明响应信息的计算开销与通信代价。并使用随机掩饰码技术确保第三方审计者不能够从完整性验证过程中获取数据块内容信息，从而实现用户数据的隐私保护。方案扩展支持同时处理批量审计任务。此两个方案皆可通过严格的安全证明来证明其安全性，并且验证过程都不含双线性对操作，因此这两个完整性验证方案皆具有轻量级和高效率的特性，适用于分布式传感器网络环境。

2. 支持用户可撤销的云数据完整性验证方案研究

提出了两个高效的支持用户可撤销的云存储数据完整性验证方案。在详细分析与描述由于用户撤销引起的云存储数据完整性验证问题的基础上，提出了用户可撤销云存储数据完整性验证的形式化模型，并利用聚合签名技术提出了 EDRPA 方案。方案有效地实现了当前用户对在云上存储的所有历史用户的数据的完整性验证，满足了由于当前用户离职而产生的数据移交需求。同时云用户可以委托第三方审计者执行安全审计任务。该方案在随机预言机模型下被证明是安全的，可以抵制由于已撤销用户和云服务商合谋而产生的攻击。

为进一步提高支持用户可撤销的云存储数据完整性验证方案的验证效率，增强其合谋抵抗能力，根据 Wang 在文献[49]中明确提出的下一步工作的思路，基于

单向代理重签名技术提出了一个具有隐私保护功能的支持用户可撤销的云存储数据公开完整性验证新方案 PRRPA。方案中所采用的单向代理重签名算法，其代理重签名密钥由当前用户私钥结合已撤销用户公钥生成，不存在私钥泄露问题，这样能够安全的实现数据所有权的转移。该方案同样在随机预言机模型下被证明是安全的，可以抵制由于已撤销用户和云服务商合谋而产生的攻击。性能分析表明，此方案在增加新功能的基础上，其完整性验证过程中通信开销与计算代价仍全部低于文献[49]中的 Panda 方案与前文的 EDRPA 方案。

3. 支持密钥更新的高效云数据完整性方案研究

设计了一个支持密钥更新的高效云存储数据完整性验证方案。方案中当用户的密钥到期需要更新时，不再需要从云端下载他的数据。而是，从云端下载一个与整个文件相比非常小的认证作为替代。这样用户可以有效地更改验证标签，从而大大地降低了通信和计算代价。同时为确保方案的安全性与高效性，提出了“零知识数据隐私”的形式化安全模型。在对 Shacham-Waters 协议^[25]改进的基础上利用简单的离散对数零知识证明使得方案在密钥更新环境中不会泄漏任何外包数据的内容信息给验证者。方案同时证明了新的安全模型，在随机预言机模型下是可靠的。通过实例评估框架的性能，实现结果与理论分析相吻合，表明新方案是实用的。

4. 对密文类型可修改的云数据可用性研究

设计了一个密文类型可修改的基于身份代理重加密方案。该方案在具有传统代理重加密方案的核心功能的同时，密文的拥有者还可以随时修改密文的类型信息。与 Liu 等研究工作相比较，方案弥补了 2 个安全漏洞，即：1、类型修改缺乏验证，攻击者可以随意修改类型标记；2、类型修改引起了新的条件性选择明文攻击问题。

7.2 未来工作展望

本文根据云存储现实应用针对云数据的完整性与可用性进行了研究，并构造了一些高效的、支持用户可撤销、高效的密钥更新的云存储数据完整性验证方案，以及密文类型修改可用性方案。但现实应用是复杂的，要确实做到云数据的安全可用，还需要进行不断地深入研究，未来值得继续关注和研究的内容主要包括：

1. 进一步提高云数据完整性验证的效率。本文提出的高效云存储数据完整性验证方案，主要是通过使用无双线性对运算的验证方案来减少方案的通信代价与计算开销，以及通过抽样验证、批量验证任务处理等方法与技术来提高完整性验

证的效率。但是，实际的云存储应用是复杂的，在不同应用背景下需要考虑的功能需求不尽相同。如何在满足不同功能需求的前提下，兼顾安全与效率是一个值得不断深入研究的课题。

2. 多用户背景下用户可撤销、密钥更新云数据完整性验证。本文提出支持用户撤销与密钥更新的完整性验证方案，并不支持群组中的用户撤销与密钥更新。因此考虑到更广泛的应用场景，将其扩展到群组用户的动态撤销与密钥更新是有必要的。

3. 扩展支持密钥更新的云数据完整性验证方案具备抗密钥泄露功能。PKI 中的关键问题是处理用户密钥泄漏或失效的问题。尽管本论文考虑了云数据密钥更新问题，但没有考虑密钥泄露的抵抗问题。抗密钥泄露的支持密钥更新的云数据完整性验证研究是下一步的主要研究工作之一。此外，论文方案中通过改进 Shacham-Waters 完整性验证方案，尽可能减少了验证过程中的双线性对操作。但是方案的整个验证过程仍然存在两个双线性对操作，验证效率仍待进一步提高。

4. 对云端密文数据的编码可转换研究。本论文使用代理重加密技术实现了云端数据密文类型的可修改，使用代理重签名技术实现了云端数据所有权的转移。同样可使用代理对数据进行重编码，然后再对重编码数据进行完整性验证。例如，由于云服务商使用的云平台不同，其确保数据存储可靠性而使用的冗余编码也可能不同。怎样在“零知识数据隐私”安全模型下保护用户隐私，进行代理重编码并对冗余编码数据进行完整性验证，这也是一值得深入研究的课题。

由于受到作者的学识水平限制，论文中难免存在不足之处，敬请各位评审专家和读者批评指正！

致 谢

时光飞逝，转眼间我即将完成博士学业，回首攻读博士学位的艰辛时光，六年征途，往昔历历在目。

首先，我要衷心感谢我的导师许春香教授。许老师的严格要求与悉心教导促使我在科研道路上留下了坚实脚印，使我的研究和论文得以顺利完成。许老师知识渊博、治学严谨、诲人不倦，实事求是的工作作风和平易近人的人格魅力都使我受益匪浅，我的点滴成就和进步都凝结了许老师的心血，在此我向许老师送上最诚挚的敬意和衷心的感谢！

其次，感谢实验室的全体成员在此期间对我的帮助，特别是张晓均博士、邓江博士、吴淮博士、曾福庚博士、张源博士、孙丽雪博士、赵继宁博士、杨兴春博士、金春花博士、谢润博士、蒋林智博士。感谢李万鹏、周让、徐辰福、薛婧婷、何瑜等硕士师弟师妹。感谢 2010 级同班博士同学的关心与帮助。谢谢你们与我在学术科研之路风雨前行，一路相伴，与你们为友，是我人生最大的财富，愿你们今后人生一帆风顺，心想事成！

我还要真诚地感谢我的领导同事在我攻博期间给予了大量的支持与帮助。

感谢父母对我养育之恩，感谢美丽贤惠的妻子席凤女士多年来在背后默默地付出，感谢张钰凯小朋友懂事听话让我能够专心科研。

最后，衷心感谢为评阅本论文而付出辛勤劳动的各位专家和学者！

参考文献

- [1] P. Mell, T. Grance. The NIST definition of cloud computing[J]. National Institute of Standards and Technology, 2009, 53(6): 50
- [2] M. Armbrust,A. Fox, R. Griffith, et al.A view of cloud computing[J].Communications of the ACM,2010, 53(4): 50-58
- [3] T. Velte, A. Velte, R. Elsenpeter. Cloud Computing, A Practical Approach[J]. Spatial Science, 2015, 60(1):197-198
- [4] J. Kincaid. Mediamax/helinkup close its doors[EB/OL]. <http://techcrunch.com/2008/07/10/mediamaxthelinkup-closes-its-doors/>, July 10, 2008
- [5] Amazon.com. Amazon s3 Availability Events[EB/OL]. <http://status.aws.amazon.com/s3-20080720.html>, July07, 2008
- [6] CloudSecurity Alliance. Top threats to Cloud Computing[EB/OL].<http://www.cloudsecurityalliance.org>, 2010
- [7] V. Kher, Y. Kim. Securing distributed storage: challenges, techniques, and systems[C]. ACM Workshop on Storage Security & Survivability, 2005, New York, USA, 9-25
- [8] B. Schroeder, G. A. Gibson. Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you?[C]. 5th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, 2007, 1-16
- [9] 秦志光,吴世坤, 熊虎. 云存储服务中数据完整性审计方案综述[J].信息网络安全, 2014(7):1-6
- [10] R. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-keycrypto systems[J]. Communications of the ACM, 1978, 21(2):120-126
- [11] R. Rivest. The MD5 Message-Digest Algorithm[J]. Ietf Rfc, 1992, 473(10):492-492
- [12] A.Muthitacharoen, R. Morris, T. M. Gil, et al.. Ivy: A Read/Write Peer-To-Peer File System[J]. Proc of the Fifth Symposium on Operating Systems Design & Implementation, 2012, 36(1):31-44
- [13] M.Kallahalla, E. Riedel, R. Swaminathan, et al.. Plutus: Scalable Secure File Sharing on Untrusted Storage[C]. Usenix Conference on File and Storage Technologies, Berkeley, CA, USA, 2003, 29-42
- [14] J.Li, M. Krohn, D. Mazieres, Secure untrusted data repository (SUNDR)[C]. Conference on Symposium on Opearting Systems Design & Implementation. Berkeley, CA, US, 2004, 9-9

- [15] A. R. Yumerefendi, J. S. Chase. Strong accountability for network storage[J]. *AcmTransact- ions on Storage*, 2007, 3(3):6-6
- [16] U. Maheshwari, R. Vingralek, W. Shapiro. How to Build a Trusted Database System on Untrusted Storage.[C]. Proceedings of the 4th Conference on Symposium on Operating System Design and Implementation, San Diego, Calif, USA, 2000, 135-150
- [17] Q. Wang, C. Wang, J. Li, et al.. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing[C]. 14th European Symposium on Research in Computer SecuritySaint-Malo, France, 2009, 355-370
- [18] E. Discovery, I. A. Management. Security guidance for critical areas of focus incloudcomputing[J]. *Evidence & Cloud Computing the Vmi Approach* Poisel Malzer & Tjoa, 2011
- [19] D Johnson, A Menezes, S Vanstone. The elliptic curve digital signaturealgorithm (ECDSA) [J]. *International Journal of Information Security*, 2001, 1(1): 36-63
- [20] D. Chaum, E. V. Heyst. Group Signatures[J]. *Lecture Notes in Computer Science*, 1991, 547(1): 257-265
- [21] A. Fiat, M. Naor. Broadcast Encryption[J].*Lecture Notes in Computer Science*, 1993, 35(8):57-63
- [22] A. Fiat, M. Naor. Broadcast encryption[C].Proceedings of the 13st Annual International Cryptology Conference (Crypto 1993), Santa Barbara, California, USA, 1993, 480-491
- [23] A. Juels, S. Burton, J. Kaliski. Pors: proofs of retrievability for large files[C]. Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS2007, Alexandria, Virginia, USA, 2007, 584–597
- [24] G. Ateniese, R. C. Burns, R. Curtmola, et al.. Provable data possession at untrusted stores[C]. Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS2007, Alexandria, Virginia, USA, 2007, 598-609
- [25] H. Shacham, B. Waters. Compact proofs of retrievability[C].*Advances in Cryptology- ASIACRYPT 2008*, Melbourne, Australia, 2008, 90-107
- [26] C. Wang, S. S. M. Chow, Q.Wang, et al.. Privacy-preserving public auditing for secure cloud storage[J]. *IEEE Transactions on Computers*, 2013, 62(2): 362–375
- [27] K. Draou, N. Bellakhal, B. G. Cheron, et al. Ensuring Data Storage Security in Cloud Computing.[J]. *Materials Research Bulletin*, 1998, 33(7):1117–1128
- [28] C. Wang, Q. Wang, K. Ren, et al.. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing[J]. *Proceedings - IEEE INFOCOM*, 2010, 62(2): 525-533

- [29] Q. Wang, C. Wang, K. Ren, et al.. Enabling public auditability and data dynamics for storage security in cloud computing[J]. IEEE Transactions on Parallel Distributed. Systems, 2011, 22(5): 847–859
- [30] J. Yuan, S. Yu. Efficient public integrity checking for cloud datasharing with multi-user modification[C]. Proceedings of International Conference on Computer Communications, INFOCOM2014, Toronto, Canada, 2014, 2121-2129
- [31] B.Wang, B. Li, H. Li. Public Auditing for Shared Data with Efficient User Revocation in the Cloud[C].the 32nd IEEE International Conference on Computer Communications 2013, 2904–2912
- [32] S. G. Worku, C. Xu, J. Zhao, et al..Secure and efficient privacy-preserving public auditing scheme for cloud storage[J]. Computers & Electrical Engineering, 2014, 40(5):1703–1713
- [33] C. Xu, Y. Zhang, Y. Yu, et al.. An efficient provable secure public auditing scheme for cloud storage[J]. KSII Transactions on Internet & Information Systems, 2014, 8(11):4226-4241
- [34] R. C. Merkle. Protocols for public key cryptosystems[C]. IEEE Symposium on Security and Privacy, Oakland, CA, USA, 1980, 122-134
- [35] C. Xu, X. He, D. Weldemariam. Cryptanalysis of Wang's auditing protocol for data storage security in cloud computing[C]. Proceedings of ICICA 2012, Chengdu, China, 2012, 422-428
- [36] Y. Zhu, H. Hu, G.J. Ahn, et al. Cooperative provable data possession for integrity verification in multicloud storage[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(12): 2231-2244
- [37] Y. Zhu, H. Wang, Z. Hu, et al. Dynamic audit services for integrity verification of outsourced storages in clouds[C]. Proceedings of the 2011 ACM Symposium on Applied Computing. New York, USA, 2011, 1550-1557
- [38] K. Yang, X. Jia. An efficient and secure dynamic auditing protocol for data storage in cloud computing[J]. IEEE Transactions on Parallel Distributed. Systems, 2013, 24(9): 1717–1726
- [39] Y. Zhu, G.J. Ahn, H. Hu, et al.. Dynamic audit services for outsourced storages in clouds[J]. IEEE Transactions on Services Computing, 2013, 6(2): 227–238
- [40] C. C. Erway, A. Küpcü, C. Papamanthou, et al.. Dynamicprovable data possession[C]. Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, 2009, 213-222
- [41] C. Wang, K. Ren, W. Lou, et al. Toward publicly auditable secure cloud data storage services[J]. IEEE Network, 2010, 24(4): 19-24

- [42] C. Wang, Q. Wang, K. Ren, et al. Toward Secure and Dependable Storage Services in Cloud Computing[J]. IEEE Transactions on Services Computing, 2013, 5(2): 220-232
- [43] Y. Zhu, S.B. Wang, H. Hu, et al. Secure collaborative integrity verification for hybrid cloud environments[J]. International Journal of Cooperative Information Systems, 2012, 21(3): 165-197
- [44] Y. Zhu, H. Hu, J. G. Ahn, et al. Efficient audit service outsourcing for data integrity in clouds[J]. Journal of Systems & Software, 2012, 85(5): 1083-1095
- [45] J. Katz, A. Sahai, B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products[C]. Proceedings of Workshop on the Theory and Application of Cryptographic Techniques (Eurocrypt 2008), Istanbul, Turkey, 2008, 146-162
- [46] B. Wang, B. Li, H. Li. Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud[M]. Springer Berlin Heidelberg, 2012, 507-525
- [47] B. Wang, H. Li, M. Li. Privacy-preserving public auditing for shared cloud data supporting group dynamics[C]. Proceedings of 2013 IEEE International Conference on Communications, Piscataway, NJ, 2013, 1946-1950
- [48] B. Wang, B. Li, H. Li. Oruta: privacy-preserving public auditing for shared data in the cloud[J]. IEEE Transactions on Cloud Computing, 2014, 2(1): 43-56
- [49] B. Wang, B. Li. H. Li. Panda: public auditing for shared data with efficient user revocation in the cloud[J]. IEEE Transactions on Services Computing, 2015, 8(1): 92-106
- [50] B. Libert, D. Vergnaud. Multi-use unidirectional proxy re-signatures[C]. Proceedings of the 15th ACM conference on Computer and communications security, New York, ACM, 2008, 511-520
- [51] M. Blaze, G. Bleumer, M. Strauss, Divertible protocols and atomic proxy cryptography[C]. International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, 1998, 127-144
- [52] I. Luan, Q. Tang, P. Hartel, et al. A Type-and-Identity-Based Proxy Re-encryption Scheme and Its Application in Healthcare[M]. Springer Berlin Heidelberg, 2008, 185-198
- [53] L. Ibraimi, Q. Tang, P. Hartel, et al. Exploring Type-and-Identity-Based Proxy Re-Encryption Scheme to Securely Manage Personal Health Record[J]. International Journal of Computational Models & Algorithms in Medicine, 2010, 1(2): 1-21
- [54] M. Green, G. Ateniese. Identity-Based Proxy Re-encryption[M]. Springer Berlin Heidelberg, 2007, 288-306

- [55] C. K. Chu, W. G. Tzeng. Identity-Based Proxy Re-encryption Without Random Oracles[C]. Proceedings of the 10th international conference on Information Security, Chile, 2007, 189-202
- [56] T. Matsuo. Proxy Re-encryption Systems for Identity-Based Encryption[M]. Springer Berlin Heidelberg, 2007, 247-267
- [57] B. Libert, D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption[J]. Information Theory IEEE Transactions on, 2011, 57(3): 1786-1802
- [58] Z. Y. Liu, G.H. Cui. A Dynamic Type and Identity-Based Proxy Re-Encryption Scheme[J]. Journal of University of Electronic Science & Technology of China, 2014, 43(3): 408-412
- [59] U.M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms[C]. Advances in cryptology-CRYPTO'94, Santa Barbara, California, USA, 1994, 271-281
- [60] D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing[C]. Proceedings of the 21st Annual International Cryptology Conference (Crypto 2001), SantaBarbara, California, USA, 2001, 213-229
- [61] D. Galindo, Boneh-Franklin. identity based encryption revisited[C].Proceedings of32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), Lisbon, Portugal, 2005, 791-802
- [62] 张先红. 数字签名原理及技术[M]. 北京：机械工业出版社, 2004: 29-37
- [63] J.M. Pollard. Monte Carlo methods for index computation (mod p) [J]. Mathematics of Computation, 1978, 32(143): 918-924
- [64] S.C. Pohlig, M.E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.)[J]. IEEE Transactions on Information Theory, 1978, 24(1): 106-110
- [65] A.Joux, K. Nguyen. Separating decision Diffie–Hellman from computational Diffie–Hellman in cryptographic groups[J]. Journal of cryptology, 2003, 16(4): 239-247
- [66] J.H.Cheon, D.H.Lee. Diffie-Hellman problems and bilinear maps[EB/OL].<http://eprint.iacr.org/2002/117>, August11, 2002
- [67] R. Barua, R. Dutta, P. Sarkar. Provably secure authenticated tree based group key agreement protocol using pairing[C].the 6th International Conference on Informationand Communications Security(ICICS'04), Malaga, Spain, 2004, 92-104
- [68] D. Boneh, E. Goh, K. Nissim. Evaluating 2-DNF formulas on ciphertexts[C]. Proceeding of 2nd Theory of Cryptography Conference (TCC 2005) , Cambridge, USA, 2005, 325-341

- [69] D. Boneh, X. Boyen, E. J Goh. Hierarchical identity based encryption with constant size ciphertext[C]. Proceedings of Workshop on the Theory and Application of Cryptographic Techniques (Eurocrypt 2005), Aarhus, Denmark, 2005, 440-456
- [70] K. Ren, C. Wang, and Q. Wang, Security Challenges for the Public Cloud[J]. IEEE Internet Computing, 2012, 16(1):69-73
- [71] N. Cao, S. Yu, Z. Yang, et al. LT codes-based secure and reliable cloud storage service[J]. Proceedings - IEEE INFOCOM, 2012, 131(5):693-701
- [72] S. R. Tate, R. Vishwanathan, L. Everhart. Multi-user dynamic proofs of data possession using trusted hardware[C]. Proceedings of the third ACM conference on Data and application security and privacy, New York, USA, 2013, 353-364
- [73] J. Yuanand, S. Yu. Proofs of retrievability with public verifiability and constant communicate on cost in cloud[C]. Proceedings of the 2013 International Workshop on Security in Cloud Computing, Hangzhou, China, 2013, 19-26
- [74] H. Wang. Proxy provable data possession in public clouds[J]. IEEE Transactions on Services Computing, 2013, 6(4): 551-559
- [75] S. Agrawal, B. Dan. Homomorphic MACs: MAC-Based Integrity for Network Coding[C]. 7th International Conference, Paris-Rocquencourt, France, 2009, 292-305
- [76] D. Chaum. Blind signatures for untraceable payments[C]. Advances in Cryptology - Proceedings of CRYPTO '82, Santa Barbara, California, USA, 1982, 199-203
- [77] A. Shamir. How to share a secret[J]. Communications of the ACM, 1979, 22(11):612-613
- [78] M. Li, N. Cao, S. Yu, and W. Lou, FindU: Private-Preserving Personal Profile Matching in Mobile Social Networks, in: Proceedings of IEEE INFOCOM 2011, 2435-2443
- [79] Ateniese G, Fu K, Green M, et al. Improved proxy re-encryption schemes with applications to secure distributed storage[J]. Acm Transactions on Information & System Security, 2005, 9(1):29-43
- [80] D. Boneh, X. Boyen, H. Shacham. Short group signatures[C]. Advances in Cryptology - CRYPTO 2004, Santa Barbara, California, USA, 2004, 41-55
- [81] X. Liang. Research on attribute-based cryptosystem[D]. Shanghai: Shanghai Jiao Tong University, 2009
- [82] 孙淑玲. 应用密码学[M]. 北京: 清华大学出版社, 2004: 43-44
- [83] T. H. Cormen, C. E. Leiserson, et al. Introduction to algorithms (second edition)[M]. Cambridge: The Massachusetts Institute of Technology Press, 2001

- [84] W. Mao. Modern cryptography: theory and practice[M].United States: Prentice Hall Professional Technical Reference, 2003
- [85] S. Goldwasser, S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information[C]. Proceedings of the 14th ACM Symposium on Theory of Computing, San Francisco, 1982, 365-377
- [86] A. Menezes, P. Van Oorschot, S. Vanstone. Handbook of Applied Cryptography[M]. Florida: The Chemical Rubber CompanyPress, 1996, 1270-1340
- [87] A. Fiat, A. Shamir. How to prove yourself: Practical solutions to identification and signature problems[C]. Advances in Cryptology-CRYPTO 1986, Santa Barbara, 1986, 186-194
- [88] M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols[C]. Proceedings of the 1st ACM conference on Computer and communications security, Fairfax, Virginia, USA, 1993, 62-73
- [89] R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology revisited[J]. Journal of the ACM (JACM), 2004, 51(4): 557-594
- [90] D. Boneh, X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles[C]. Advances in Cryptology-EUROCRYPT 2004, Interlaken, Switzerland, 2004, 223-238
- [91] D. Pointcheval. Asymmetric cryptography and practical security[J]. Journal of Telecommunications and Information Technology, 2002, 4(2): 41-56
- [92] N. Subramanian, C. Yang, W. Zhang. Securing distributed data storage and retrieval in sensor networks[J]. Pervasive & Mobile Computing, 2007, 3(6):659-676
- [93] H. Luo, F. Ye, J. Cheng, et al. A two-tier data dissemination model for large-scale wireless sensor networks[C]. Proceedings of the 8th annual international conference on Mobile computing and networking, Bristol, USA, 2002, 148-159
- [94] G. Wang, G. Cao, T. La Porta, et al. Sensor relocation in mobile sensor networks[J]. Proc of IEEE Infocom, 2005, 4: 2302-2312
- [95] E. Mykletun, J. Girao, D. Westhoff. Public key based cryptoschemes for data concealment in wireless sensor networks[C]. Proceedings of the IEEE International Conference on Communications, Istanbul, 2006, 2288-2295
- [96] J. Girao, D. Westhoff, E. Mykletun, et al. TinyPEDS: Tiny persistent encrypted data storage in asynchronous wireless sensor networks[J]. Ad Hoc Networks, 2007, 5(7): 1073-1089

- [97] T. S. J. Schwarz, E.L. Miller.: Using Algebraic Signatures to Check Remotely Administered Storage[J]. 26th IEEE International Conference on Distributed Computing Systems, 2006, 26(12): 12
- [98] S. Yu, C. Wang, K. Ren, et al. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing[J]. Proceedings - IEEE INFOCOM, 2010, 29(16): 1-9
- [99] M. Li, S. Yu, K. Ren, et al. Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings[C], 6th International ICST Conference, Singapore, 2010: 89-106
- [100] M. A. Shah, M. Baker, J. C. Mogul, et al. Auditing to Keep Online Storage Services Honest[J]. Proc.of Hotos Xi.usenix, 2007
- [101] M. Arrington, Gmail Disaster: reports of mass email deletions.<http://www.techcrunch.com/2006/12/28/gmail-disaster-reports-of-mass-email-deletions> [EB/OL]. December28, 2006
- [102] M. A. Shah, R. Swaminathan, M. Baker. Privacy-Preserving Audit and Extraction of Digital Contents[J]. Pasos Revista De Turismo Y Patrimonio Cultural, 2008, 477-494
- [103] A. L. Ferrara, M. Green, S. Hohenberger, et al.. Practical short signature batch verification[C]. The Cryptographers'Track at the RSA Conference 2009, San Francisco, CA, USA, 2009, 309-324
- [104] G. Ateniese, R. D. Pietro, L.V. Mancini, et al.. Scalable and efficient provable data possession[C]. Proceedings of the 4th international conference on Security and privacy in communication netowrks, Istanbul, Turkey, 2008, 9-18
- [105] G. Ateniese, S. Hohenberger. Proxy re-signatures: new definitions, algorithms, and applications[C].Proceedings of the 12th ACM conference on Computer and communications security, Alexandria, VA, USA, 2005, 310–319
- [106] C. Delerablée, P. Paillier, D. Pointcheval. Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Key[M]. Springer Berlin Heidelberg, 2007, 39-59
- [107] Q. Liu, G. Wang, J. Wu. Efficient Sharing of Secure Cloud Storage Services[C]. IEEE International Conference on Computer & Information Technology, Bradford, 2010, 922-929
- [108] Q. Liu, G. Wang, J. Wu. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment[J]. Information Sciences An International Journal, 2014, 258(3): 355–370
- [109] X. Zhang, C. Xu, X. Zhang. Efficient pairing-free privacy-preserving auditing scheme for cloud storage in distributed sensor networks[J]. International Journal of Distributed Sensor Networks, 2015, 2015: 1-10

- [110] M. Raju, B. Lanitha. Survey about Cloud Computing Threats[J]. International Journal of Computer Science and Information Technologies, 2014, 5(1): 384-389
- [111] D. Gay, P. Levis, R. V. Behren, et al.. The nesC language: A holistic approach to networked embedded systems[J]. Acm Sigplan Notices, 2003, 38(4): 1-11
- [112] N. Aminzadeh, Z. Sanaei, S. H. A. Hamid. Mobile storage augmentation in mobile cloud computing: Taxonomy, approaches, and open issues[J]. Simulation Modelling Practice & Theory, 2014, 50:96-108
- [113] M. Xie, H. Wang, J. Yin, et al. Integrity Auditing of Outsourced Data[J]. In VLDB, 2007, 782-793
- [114] Q. Wang, C. Wang, K. Ren, et al. Enabling public auditability and data dynamics for storage security in cloud computing[C]. Proceeding of ESORICS 2009, Saint Malo, France, 2009, 21-25
- [115] Y. Yu, H. A. Man, Y. Mu, et al. Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage[J]. International Journal of Information Security, 2015, 14(4): 307-318
- [116] P. Barreto, M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order[C]. Selected Areas in Cryptography: 12th International Workshop, SAC, 2005, Kingston, ON, Canada, 2005, 319-331
- [117] B. Lynn. The Pairing-Based Cryptography Library (0.5.13)[EB/OL]. <http://crypto.stanford.edu/pbc/>, February 2007
- [118] A. Kate. The Pairing-Based Cryptography (PBC) Library - C++ Wrapper Classes (0.8.0)[EB/OL]. <http://crysph.uwaterloo.ca/software/PBCWrapper/>
- [119] S. D. Sapatnekar, P. Kanase. Implementation of Public Cloud Data Auditing with Practical Key-Updating and Zero Knowledge Privacy[J]. Journal of Engineering Research and Applications, 2015, 5(7): 46-51
- [120] G. Ateniese, R. Burns, R. Curtmola, et al. Remote data checking using provable data possession[J]. Acm Transactions on Information & System Security, 2011, 14(1): 1165-1182

攻读博士学位期间取得的成果

1. 学术科研论文发表情况

- [1] **Xinpeng Zhang**, Chunxiang Xu, Xiaojun Zhang. Efficient pairing-free privacy-preserving auditing scheme for cloud storage in distributed sensor networks [J]. International Journal of Distributed Sensor Networks, 2015, ID: 593759. (SCI, IF: 0.665, IDS: CO1JH), WOS: 000358910100001
- [2] **Xinpeng Zhang**, Chunxiang Xu, Xiaojun Zhang, Wei Sai, Ying Li, Tao Zhou. New data storage Auditing Protocols [C]. 2nd International Conference on Teaching and Computational Science (ICTCS), Shenzhen, 2014, 40-43. (CPCI, WOS: 000346248300010)
- [3] 张新鹏, 许春香, 张晓均, 邓江, 黄新. 对类型可修改的基于身份代理重加密方案的改进[J]. 电子科技大学报, 稿件编号 2015103. (EI, 已录用)
- [4] **Xinpeng Zhang**, Chunxiang Xu, Xiaojun Zhang, Taizong GuWei, Zhi Geng, Guoping Liu. Efficient dynamic integrity verification for big data supporting users revocability [J]. Information, ID: information-119914. (EI, 已录用)
- [5] 张新鹏, 许春香, 张新颜, 赛伟, 韩兴阳, 刘国平. 基于代理重签名的支持用户可撤销的云存储数据公共审计方案. 计算机应用研究(核心, 已录用)
- [6] Qianqi Le, Guowu Yang, William N. N. Hung, **Xinpeng Zhang**, Fuyou Fan. A multiobjective scatter search algorithm for fault-tolerant NoC mapping optimisation. International Journal of Electronics[J]. 2014,101(8),1056-1073. (SCI: AJ4XB)
- [7] Qianqi Le, Guowu Yang, William N. N. Hung, Xiaoyu Song, **Xinpeng Zhang**. Pareto optimal mapping for tile-based network-on-chip under reliability constraints. International Journal of Computer Mathematics[J].Accepted. (SCI)
- [8] Fugeng Zeng, Chunxiang Xu, **Xinpeng Zhang**. Security analysis for a ciphertext-policy attribute- based encryption scheme[J]. Advances in Information Sciences and Service Sciences, 2012, 7: 201-207 (EI: 20123015280903)
- [9] Yong Yu, **Xinpeng Zhang**, Man Ho Au, Guomin Yang, Willy Susilo, Yi Mu, Zhenfei Zhang. Public Auditing with Practical Key Update and Zero Knowledge Privacy for Embedded Cloud Computing [J]. ACM Transactions on Embedded Computing Systems. (SCI, submitted)
- [10] Xiaojun Zhang, Chunxiang Xu, **Xinpeng Zhang**. Efficient privacy-preserving light-weight auditing scheme for cloud-assisted wireless body area networks [J]. SCIENCE CHINA Information Sciences. (SCI, submitted)

2. 专利申请受理情况

[1] 无双线性对的云存储数据安全完整性验证方案方法。

专利申请号：201410255769.8（第三发明人）

攻读博士学位期间参与的科研项目

[1] 国家自然科学基金项目：数据云存储中的安全完整性验证方案方法研究。

项目编号：61370203，2014.01-2017.07

[2] 保密通信重点实验室基金项目：具有自保护 XXX 技术研究。

项目编号：9140C110301110C1103，2011.07-2013.06，已结题。

[3] 总装预研基金项目：基于云计算 XXX 的仿真支撑技术。

项目编号：9140A04020311DZ02，2011.06-2013.07，已结题。

[4] 四川省重点科学技术支撑项目：军地联动快速应急卫勤保障体系研究。

项目编号：2012SZ0162，2013.01-2015.12，已结题。



博士学位论文

DOCTORAL DISSERTATION