编号： CACR2022B2KQZR

作品类别： ☑软件设计 □硬件制作 □工程实践 □密码应用技术

**2022年第七届全国密码技术竞赛作品设计报告**

题目：云存储环境中用户数据的完整性验证算法设计

2022年10月25日

中国密码学会

|  |
| --- |
| 基本信息表 |
| 编号：CACR2022B2KQZR |
| 作品题目：云存储环境中用户数据的完整性验证算法设计 |
| 作品类别：☑软件设计 □硬件制作 □工程实践 □密码技术应用 |
| 作品内容摘要：  云存储为用户提供了便捷的数据存储服务，得到了迅速发展。然而在云存储环境下，用户失去了对数据的直接控制，云服务器为了降低维护数据的成本，可能会删除掉用户不经常访问的数据。因此，为了确保用户外包数据的完整性，学者们提出了外包数据完整性验证方法。该方法无需取回外包数据，就可以验证外包数据是否完整。本参赛团队基于BLS（Boneh-Lynn-Shacham）签名和哈希函数设计了一种外包数据完整性验证算法。该算法支持数据的公开验证和私自验证，且算法能满足以下功能：（1）用户无需储存外包数据；（2）如果云服务器上数据被窜改或删除，能够被发现；（3）验证数据完整性时，通信开销和计算开销较小。 |
| 关键词（五个）：  云存储；外包数据完整性验证；数字签名；哈希函数；双线性对 |

# 1.作品功能与性能说明

云存储中的数据完整性验证[1]，可以分为公开验证和私自验证两种。一般而言，私自验证[2,3]利用对称密码算法构造，效率较高，但是由于不支持公开验证，在私自验证中，用户和云服务器之间可能发生争端，比如：用户通过验证发现外包数据被窜改了，而云存储服务器却声称数据是完整的。由于外包数据的标签是用户生成的，而生成标签的密钥只有用户自己知道，因而第三方机构无法介入解决争端。公开验证利用公钥密码算法构造，虽然效率较低，但是支持公开验证，因而不仅用户与云存储服务器之间争端容易解决，而且用户可以授权第三方审计者完成外包数据的完整性验证工作。

本作品利用BLS（Boneh-Lynn-Shacham）签名[4]和哈希函数（映射到点的哈希）来实现数据的私自验证和公开验证。可以实现以下功能：参数生成、用户参数生成、标签生成、发起挑战、响应生成、私自/公开验证。具体而言，用户首先将待上传数据分块并编号，利用签名算法，计算数据块的标签，然后将数据和标签一同上传至云服务器。在需要验证数据完整性时，用户/审计者将需要抽查的数据块编号和随机数发送给云服务器，云服务器收到验证请求后，利用被抽查数据块和相应的标签计算出证明文件并发给用户，之后用户/审计者通过相应的算法来验证证明文件的正确性，从而验证出外包数据的完整性。

此外，本作品可以高效地实现数据完整性验证，其通信开销远小于将原数据从云服务器发回至用户/审计者进行验证的开销，且计算开销较小，在私自验证时，不需要用户有强大的计算能力，就可以在很短的时间内验证数据的完整性。

# 2.设计与实现方案

本作品包含以下算法：全局参数生成（GlobeSetup）、用户参数生成（UserSetup）、标签生成（TagGen）、发起挑战（Chal）、响应生成（ProofGen）、私自/公开验证（Verify）。

全局参数生成（GlobeSetup）：该算法生成并公开全局参数。

用户参数生成（UserSetup）：给定全局公开参数，用户生成自己的公/私钥对（*pk*,*sk*）。

标签生成（TagGen）：给定全局公开参数，用户利用私钥*sk*，生成数据标签*tags*，并把文件*M*和数据标签*tags*外包到云服务器上。

发起挑战（Chal）：给定数据*M*的块数*n*，用户向云服务器发起一个挑战*chal*。

响应生成（ProofGen）：给定挑战*chal*，全局公开参数，文件*M*及其标签*tags*，云服务器返回响应*resp*给用户。

私自/公开验证（Verify）：给定全局公开参数，挑战*chal*，响应*resp*，通过相应的算法，验证外包数据的完整性。如果数据被窜改了，返回“*false*”，否则，返回“*true*”。

## 2.1 实现原理

本作品的数据完整性验证方案如下:

1. 全局参数生成（GlobeSetup）：设*p*是一个大素数，*G*和*GT*是两个*p*阶循环群，*e*：*G×G→GT*是一个双线性映射，*g*是*G*的生成元，*H*:{0,1}*\**→*G*为映射到点的哈希函数。
2. 用户参数生成（UserSetup）：用户随机选择*α*∈*Zp*，计算*v=gα*。公钥为*pk*=*v*，私钥为*sk*=*α*;
3. 标签生成（TagGen）：给定数据*M*，将*M*分割为*n*个块，再将每个块分为*s*个部分，每个文件块可以表示为：{*mij*}1≤*i*≤*n*,1≤*j*≤*s*。用户选择*s*个随机元素，计算，选择文件名*name*，得到，计算文件的标签块*σ*：



上传数据*M*和标签*tag*={*σi*}1≤*i*≤*n*至云服务器。

1. 发起挑战（Chal）：用户/审计者选择要挑战的数据，从集合[1,*n*]中随机选取一个子集*I*，对于每个*iI*，随机选取一个元素*vi*，构成一个查询集合*chal* = {(*i,vi*)}。发送*chal*给云服务器。
2. 响应生成（ProofGen）：云服务器利用数据块{*mij*}，标签块{*σi*}，和来自用户/审计者的查询集合*chal=*{(*i,vi*)}，计算：





将值*µ*1,*…*,*µs*，*σ*，返回给验证者。

1. 私自验证（PrivateVerify）：用户解析服务器的响应以获得*μ* ={*μ*1,*…*,*μs* }、*σ*和，并利用存储在本地的，检查下列等式是否成立：



如果是，输出“*true*”，表示本次验证通过；否则，输出“*false*”。

1. 公开验证（PublicVerify）：审计者解析服务器的响应以获得*μ* ={*μ*1,*…*,*μs* }、*σ*和。检查下列等式是否成立：



如果是，输出“*true*”，表示本次验证通过；否则，输出“*false*”。

## 2.2 性能分析

我们对比了Shacham 等[5]方案与本作品的计算开销（表1）。在表1中，H、A、M、P和E分别表示哈希计算、加法、乘法、双线性配对和求幂，|*I*|表示在查询集合*chal*中所包含的挑战块的数量。在文献[5,section3.3]中，标签生成需要*n*(*s* + 1)次幂，运算在我们的作品中，只需要2*n*次的幂运算；在私自验证时，本作品幂运算的次数为2|*I*|，计算量小于公开验证。因此，对比文献[1]，本作品在标签生成和私自验证中计算速度有一定优势。

|  |  |  |
| --- | --- | --- |
|  | 文献[5,section3.3] | 本作品 |
| 标签生成 | *n*H+*ns*M+*n*(*s* + 1)E | *n*H+*ns*M+*n*(*s* - 1)A+2*n*E |
| 响应生成 | ((*s* + 1)*|I| -* 1)M + *|I|E* + *s*(|*I*| - 1)A | ((*s*+1)*|I| -*1)M + *|I|E* + *s*(|*I*| - 1)A |
| 公开验证 | 2P + |*I*|H + (|*I*| + *s* - 1)M + (|*I*| + *s*) E | 2P + |*I*|H + (|*I*| + *s* - 1)M + (|*I*| + *s*) E |
| 私自验证 | - | 2P + |*I*|H + *s*|*I*| M + (*s* - 1) |*I*|A+2|*I*|E |

表1 计算开销对比

# 3.系统测试与结果

测试环境：XiaoXinPro 16ACH 2021（处理器: AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz、内存: 16.0 GB、操作系统：Windows11），使用IntelliJ IDEA开发工具并使用JAVA语言（java version :1.8.0\_281）和JPBC库进行实验，选取Type A类型素数阶椭圆曲线。

## 3.1测试方案

我们在本地计算机上模拟数据在云端的存储方式，假设由用户上传数据和相应的标签块至云服务器，然后由用户（私自验证）发起完整性验证挑战，测试外包数据遭到窜改、删除时的验证结果，并测试在验证不同文件大小时标签生成的时间以及验证所花费的时间。

## 3.2 功能测试

假设用户需要外包的数据为testfile.rar，其大小为313,844,087 Byte（图1）。

|  |
| --- |
| 图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成 |
| 图1 测试的数据 |

接下来对数据分块并且编号，每个块的大小为1,048,576 Byte（图2、图3、图4）。在计算标签块时，再将每个块分成20Byte的小块。根据我们的算法，每1,048,576 Byte大小的数据块对应一个标签*tag*，每个标签块的大小为128Byte（图5、图6、图7）。

|  |  |
| --- | --- |
| 图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成 | 图形用户界面, 文本, 应用程序  描述已自动生成 |
| 图2 每个数据块的大小 | 图3 分块后的数据文件夹 |
| 表格  中度可信度描述已自动生成 |  |
| 图4分块后的数据 | 图5 生成的标签tag大小 |
| 表格  描述已自动生成 | 图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成 |
| 图6 生成的标签tag文件夹 | 图7 每个标签块的大小 |

模拟将数据块和标签上传到云服务器，由用户发起验证请求（假设验证所有文件块），并将验证结果以及所用时间输出（图8）。

|  |
| --- |
|  |
| 图8 验证结果 |

模拟服务器删除部分数据块时的运行结果，假设删除编号为5的文件块（图9、图10）。

|  |
| --- |
| 表格  中度可信度描述已自动生成 |
| 图9 删除编号为5的数据块 |
| 文本  描述已自动生成 |
| 图10 删除数据块时的验证结果 |

模拟云服务器窜改数据时的验证结果（图11）。

|  |
| --- |
| 文本  描述已自动生成 |
| 图11 窜改数据块时的验证结果 |

## 3.3 性能测试

我们测试了数据从5MB到200MB大小下标签块的生成时间，以及用户验证（私自验证）过程中各阶段所花费的时间和通信开销（假设用户验证所有文件块）。

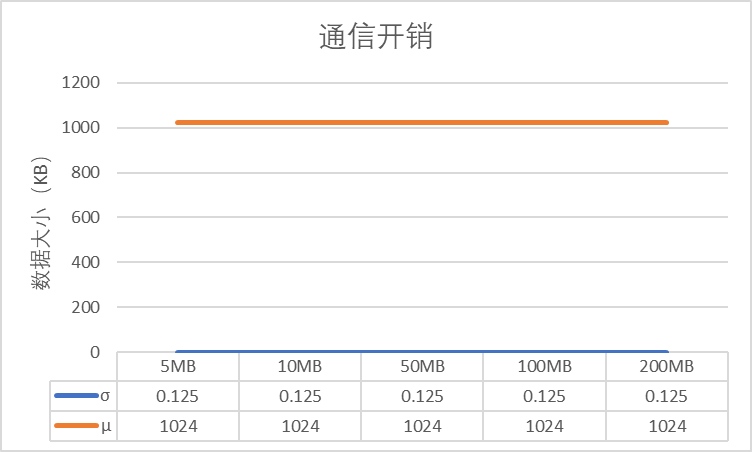
从图12可以看出，标签tag生成的时间随数据的大小增加而增加。在验证过程中，服务器计算σ和μ的时间随着用户验证的数据块的数量增加而增加。在用户计算阶段，验证的计算时间不随数据的大小而变化，并且能够在20ms之内完成验证计算。

|  |
| --- |
|  |
| 图12 验证计算时间 |

从图13可以看出，标签块tag所占的空间随着数据大小的变化而变化。本算法中，每1,048,576 Byte（1MB）大小的数据对应一个标签块，每个标签块的大小为128Byte，远小于数据大小。换言之，标签不会占用过多的存储空间。

从图14可以看出，用户验证时，云服务器向用户发送的证明数据大小*σ*和*μ*不会随着外包数据的大小变化而变化。在用户的外包数据越大时，本算法用户与云服务器之间的通信开销的优势越明显。

|  |
| --- |
|  |
| 图13 标签tags存储空间 |



|  |
| --- |
|  |
| 图14 通信开销 |

# 4.应用前景

数据完整性审计是保证云存储外包数据完整性的重要手段之一。随着越来越多的企业或者用户将数据存储在云端，保证数据的安全性问题就愈发重要。对于使用云存储服务的用户来说，在没有本地数据备份的情况下，如何找到一种高效可行的方法，来对所存储的外包数据进行完整性验证。从信息技术长远发展来看，云服务产品能够对用户存储在云端的数据提供高效的可证明的数据完整性验证机制是十分有意义的，也是云计算能够得到广泛应用的前提。

# 5. 结论

本作品实现了一种数据的完整性验证算法，支持数据的公开验证和私自验证。经过分析和测试，我们的作品对于外包数据的完整性验证是高效的，在私自验证中，用户能够在20ms内完成数据的验证计算，且具有较低的通信开销，能够很好地完成在云存储环境中用户数据的完整性验证。

# 6. 参考文献

[1] Mehdi Sookhak, Hamid Talebian, Ejaz Ahmed, Abdullah Gani, and Muhammad Khurram Khan. A review on remote data auditing in single cloud server: Taxonomy and open issues[J]. *J. Network and Computer Applications*, 2014, 43: 121–141.

[2] Hao Yan, Jiguo Li, Jinguang Han, and Yichen Zhang. A novel efficient remote data possession checking protocol in cloud storage[J]. IEEE *Trans. Information Forensics and Security*, 2017, 12(1): 78–88.

[3] Fei Chen, Zhipeng Li, Changkun Jiang, Tao Xiang and Yuanyuan Yang. Cloud Object Storage Synchronization: Design, Analysis, and Implementation[J]. IEEE *Transactions on Parallel and Distributed Systems*, 2022, 33(12) : 4295-4310.

[4] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pair-ing[J]. J. *Cryptology*, 2004, 17(4): 297–319.

[5] Hovav Shacham and Brent Waters. Compact proofs of retrievability[C]. In *14th* *International Conference on the Theory and Application of Cryptology and Information Security*, *ASIACRYPT 2008*, Melbourne, Australia, December 7-11, 2008:90–107.