Transparent Data Deduplication in the Cloud

云中透明的重复数据消除

Frederik Armknecht

弗雷德里克·阿姆克内赫特

University of Mannheim 68131 Mannheim, Germany armknecht@uni-mannheim.de

曼海姆大学，德国曼海姆 68131

Ghassan O. Karame

加桑·卡拉姆

NEC Laboratories Europe 69115 Heidelberg, Germany ghassan.karame@neclab.eu

NEC 实验室欧洲 69115 德国海德堡

ABSTRACT

摘要

Cloud storage providers such as Dropbox and Google drive heav-ily rely on data deduplication to save storage costs by only stor-ing one copy of each uploaded file.Although recent studies report that whole file deduplication can achieve up to 50% storage reduc-tion, users do not directly benefit from these savings—as there is no transparent relation between effective storage costs and the prices offered to the users.

Dropbox 和 Google drive 等云存储提供商非常依赖重复数据删除，只存储每个上传文件的一个副本，从而节省存储成本。尽管最近的研究报告称，全文件重复数据消除可实现高达 50%的存储缩减，但用户并未直接受益于这些节省，因为有效存储成本与提供给用户的价格之间没有透明的关系。

In this paper, we propose a novel storage solution, ClearBox, which allows a storage service provider to transparently attest to its customers the deduplication patterns of the (encrypted) data that it is storing.By doing so, ClearBox enables cloud users to ver-ify the effective storage space that their data is occupying in the cloud, and consequently to check whether they qualify for bene-fits such as price reductions, etc.ClearBox is secure against ma-licious users and a rational storage provider, and ensures that files can only be accessed by their legitimate owners.We evaluate a prototype implementation of ClearBox using both Amazon S3 and Dropbox as back-end cloud storage.Our findings show that our so-lution works with the APIs provided by existing service providers without any modifications and achieves comparable performance to existing solutions.

在本文中，我们提出了一种新的存储解决方案，即 ClearBox，它允许存储服务提供商向其客户透明地证明其存储的(加密的)数据的重复数据删除模式。通过这样做，ClearBox 使云用户能够验证他们的数据在云中占据的有效存储空间，并因此检查他们是否有资格获得诸如降价等好处。ClearBox 对合法用户和理性存储提供者来说是安全的，并且确保文件只能被合法所有者访问。我们使用亚马逊 S3 和 Dropbox 作为后端云存储来评估 ClearBox 的原型实现。我们的发现表明，我们的解决方案可以在不做任何修改的情况下与现有服务提供商提供的 API 一起工作，并获得与现有解决方案相当的性能。

Categories and Subject Descriptors

类别和主题描述符

C.2.0 [Computer-Communication Networks]: General - Secu-rity and protection.

计算机通信网络:一般安全和保护。

General Terms

泛称

Security, Measurement, Experimentation.

安全，测量，实验。

Keywords

关键词

Cloud security;Secure data deduplication;Transparent attestation of deduplication.

云安全；安全的重复数据删除；重复数据消除的透明证明。

DOI: http://dx.doi.org/10.1145/2810103.2813630.

http://dx.doi.org/10.1145/2810103.2813630.

Jens-Matthias Bohli

延斯-马提亚斯·博利

NEC Laboratories Europe 69115 Heidelberg, Germany Jens-Matthias.Bohli@neclab.eu

NEC 实验室欧洲 69115 德国海德堡延斯-马提亚斯。Bohli@neclab.eu

Franck Youssef

弗兰克·优素福

NEC Laboratories Europe 69115 Heidelberg, Germany franck.youssef@neclab.eu

NEC 实验室欧洲 69115 德国海德堡

1.INTRODUCTION

1.介绍

Cloud storage services have become an integral part of our daily lives.With more and more people operating multiple devices, cloud storage promises a convenient means for users to store, access, and seamlessly synchronize their data from multiple devices.

云存储服务已经成为我们日常生活中不可或缺的一部分。随着越来越多的人操作多种设备，云存储为用户提供了一种方便的方式来存储、访问和无缝同步来自多种设备的数据。

The increasing adoption of the cloud is also fueled by the mul-titude of competing cloud storage services which offer relatively cheap services.For example, Dropbox offers its customers free 2 GB accounts, Google drive offers 100 GB for only 1.99 USD, while Box.com offers its business clients unlimited storage for only 12 EUR per month.These competitive offers are mainly due to the sharp plummet exhibited by modern hard drives, going from 20 USD per GB to just about few cents per GB in 2014 [9].

提供相对便宜服务的多种竞争云存储服务也推动了云的日益普及。例如，Dropbox 为其客户提供免费的 2 GB 帐户，谷歌硬盘提供 100 GB，仅售 1.99 美元，而 Box.com 为其商业客户提供无限存储，每月仅售 12 欧元。这些有竞争力的优惠主要是由于现代硬盘价格大幅下跌，从每 GB 20 美元跌至 2014 年的每 GB 几美分[9]。

To further increase their profits,existing cloud storage providers adopt aggressive storage efficiency solutions when storing their clients' data.Namely, existing clouds store duplicate data (either at the block level or the file level) uploaded by different users only once—thus tremendously saving storage costs.Recent studies show that cross-user data deduplication can save storage costs by more than 50% in standard file systems [35,36], and by up to 90-95% for back-up applications [35].

为了进一步增加利润，现有云存储提供商在存储客户数据时采用了积极的存储效率解决方案。也就是说，现有的云只存储由不同用户上传一次的重复数据(块级或文件级)，从而极大地节省了存储成本。最近的研究表明，跨用户重复数据消除可以在标准文件系统中节省 50%以上的存储成本[35，36]，在备份应用程序中节省高达 90-95%的存储成本[35]。

The literature features a large number of proposals for securing data deduplication (e.g., [14, 25, 42]) in the cloud.All these pro-posals share the goal of enabling cloud providers to deduplicate encrypted data stored by their users.Such solutions allow the cloud provider to reduce its total storage, while ensuring the confidential-ity of stored data.

文献中有大量关于在云中保护重复数据消除(例如[14，25，42])的建议。所有这些解决方案都有一个共同的目标，即让云提供商能够对用户存储的加密数据进行重复数据消除。此类解决方案允许云提供商减少总存储，同时确保存储数据的机密性。

By doing so, existing solutions increase the profitability of the cloud, but do not allow users to directly benefit from the savings of deduplication over their data.Notice that cloud service providers charge their customers based on the amount of data that they store— irrespective of the level of data deduplication exhibited by their data.However, a user who is using the cloud as back-up storage should benefit—and rightly so—from reductions (by up to 90%), when compared to a client who is storing personal files in the cloud which are less likely to be deduplicated.In Figure 1, we estimate the cost reductions per user due to data deduplication in comparison to the price per user in existing commodity cloud providers such as Dropbox, Google drive, and Microsoft Onedrive.Our estimates clearly suggest that there is considerable room for price reductions for those users whose data undergoes considerable deduplication.

通过这样做，现有解决方案提高了云的盈利能力，但不允许用户直接从重复数据消除对其数据的节约中受益。请注意，云服务提供商根据其存储的数据量向其客户收费，而不管其数据显示的重复数据消除级别如何。但是，与在云中存储不太可能进行重复数据消除的个人文件的客户端相比，使用云作为备份存储的用户应该受益于数据减少(最多 90%)。在图 1 中，我们估计了与现有商品云提供商(如 Dropbox、Google drive 和微软 Onedrive)的每个用户价格相比，重复数据消除为每个用户带来的成本降低。我们的估计清楚地表明，对于那些数据经过大量重复数据消除的用户来说，有相当大的降价空间。

In this paper, we address this problem and we propose a novel secure storage solution, dubbed ClearBox, which enables a cloud provider to transparently and verifiably attest the deduplication pat-terns of every file stored at the cloud.Our solution relies on gate-

在本文中，我们解决了这个问题，并提出了一种新的安全存储解决方案，称为 ClearBox，它使云提供商能够透明且可验证地证明存储在云中的每个文件的重复数据消除模式。我们的解决方案依赖于 gate-

Cloud services are contributing to a 150 billion USD market [6].

云服务正在为 1500 亿美元的市场做出贡献[6]。

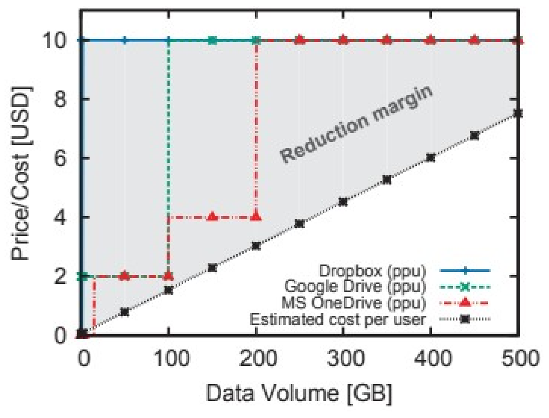


Figure 1: Cost reductions due to data deduplication vs. prices of commodity storage providers.The blue, green, and red curves show the price currently charged by Dropbox, Google Drive, and MS OneDrive, respectively.The dotted black line depicts the estimated cost of storage per user in Amazon S3 [1] after data undergoes deduplication.We assume that 50% of the data stored by clients is deduplicated [36] with the data pertain-ing to 2 other cloud users and that clients download 0.96% of the data stored in their accounts per day [34].The "reduction margin" refers to the difference between the price borne by the users and the effective cost of users' storage after deduplica-tion."ppu" refers to the price per user.

图 1:重复数据消除带来的成本降低与商品存储提供商的价格对比。蓝色、绿色和红色曲线分别显示了 Dropbox、Google Drive 和 MS OneDrive 目前的收费价格。黑色虚线描述了数据经过重复数据消除后，亚马逊 S3 [1]中每个用户的估计存储成本。我们假设客户端存储的 50%的数据已经过重复数据消除[36]，数据属于另外两个云用户，并且客户端每天下载其帐户中存储的 0.96%的数据[34]。“降价幅度”指的是用户承担的价格与用户在重复数据删除后的有效存储成本之间的差额。“ppu”指的是每个用户的价格。

way to orchestrate cross-user file-based deduplication prior to stor-ing files on (public) cloud servers.ClearBox ensures that files can only be accessed by legitimate owners, resists against a curious cloud provider, and enables cloud users to verify the effective stor-age space occupied by their encrypted files in the cloud (after dedu-plication).By doing so, ClearBox provides its users with full trans-parency on the storage savings exhibited by their data;this allows users to assess whether they are acquiring appropriate service and price reductions for their money—in spite of a rational gateway that aims at maximizing its profit.

在(公共)云服务器上存储文件之前，协调跨用户基于文件的重复数据消除的方法。ClearBox 确保文件只能由合法所有者访问，抵制好奇的云提供商，并使云用户能够验证他们的加密文件在云中(重复数据消除后)占用的有效存储空间。通过这样做，ClearBox 为其用户提供了他们的数据所展示的存储节省的完全灵活性；这使得用户能够评估他们是否为他们的钱获得了适当的服务和价格降低——尽管理性网关的目标是利润最大化。

ClearBox can be integrated with existing cloud storage providers such as Amazon S3 and Dropbox without any modifications, and motivates a new cloud pricing model which takes into account the level of deduplication undergone by data.We believe that such a model does not threaten the profitability of the cloud business and—on the contrary—gives considerable incentives for users to store large and popular data such as music and video files in the cloud (since the storage costs of popular data might be cheaper).In summary, we make the following contributions in this work:

ClearBox 可以与现有的云存储提供商(如亚马逊 S3 和 Dropbox)集成，无需任何修改，并推动了一种新的云定价模式，该模式考虑了数据所经历的重复数据删除级别。我们认为，这种模式不会威胁到云业务的盈利能力，相反，它会极大地激励用户将音乐和视频文件等大型流行数据存储在云中(因为流行数据的存储成本可能更低)。总之，我们在这项工作中做出了以下贡献:

Concrete Instantiation: We describe a cloud storage scheme, dubb-ed ClearBox, which employs a novel cryptographic tree-based accumulator, CARDIAC, to attest in logarithmic time (with respect to the number of clients that uploaded the file) the deduplication patterns of every file stored in the cloud.ClearBox additionally leverages Proofs of Ownership [23, 27], and self-expiring URL commands in order to effectively manage access control on the files stored in the cloud.

具体实例:我们描述了一个云存储方案，称为 ClearBox，它采用了一种新的基于加密树的累加器 CARDIAC，以对数时间(相对于上传文件的客户端数量)证明云中存储的每个文件的重复数据消除模式。

ClearBox 还利用所有权证明[23，27]和自动过期的 URL 命令，以便有效地管理对存储在云中的文件的访问控制。

Security Analysis: We provide a model for ClearBox and analyze

安全分析:我们为 ClearBox 提供了一个模型并进行分析

the security of our proposal according to our model.Namely, we show that ClearBox enables users to verify the dedupli-cation undergone by their files in spite of a rational provider that aims at maximizing its profit in the system.In addition, we show that ClearBox resists against malicious clients and a curious storage provider.

根据我们的模型，我们建议的安全性。也就是说，我们展示了 ClearBox 使用户能够验证他们的文件所经历的重复数据删除，尽管一个理性的提供商的目标是在系统中获得最大的利润。此外，我们展示了 ClearBox 抵抗恶意客户端和好奇的存储提供商。

Prototype Implementation: We implement and evaluate a proto-

原型实现:我们实现并评估一个原型

type based on ClearBox using both Amazon S3 and Dropbox as back-end cloud storage, and we show that our proposal does not impair the experience witnessed by cloud users and incurs tolerable overhead on the gateway when orchestrating file operations amongst its clients.

基于 ClearBox 的类型，使用亚马逊 S3 和 Dropbox 作为后端云存储，我们表明，我们的提议不会损害云用户的体验，并且在客户端之间编排文件操作时，不会在网关上产生可容忍的开销。

The remainder of this paper is organized as follows.In Sec-tion 2, we introduce our model and goals.In Section 3, we present ClearBox and analyze its security.In Section 4, we evaluate a pro-totype implementation based on the integration of ClearBox with Amazon S3 and Dropbox.In Section 5, we review related work in the area, and we conclude the paper in Section 6.In Appendices A and B, we provide additional insights with respect to our scheme.

本文的其余部分组织如下。在第二节中，我们介绍了我们的模型和目标。在第 3 节中，我们介绍了 ClearBox 并分析了它的安全性。在第 4 节中，我们评估了一个基于 ClearBox 与亚马逊 S3 和 Dropbox 集成的原型实现。在第 5 节中，我们回顾了该领域的相关工作，并在第 6 节中总结了论文。在附录 A 和 B 中，我们提供了关于我们计划的更多见解。

2.MODEL

2.模型

Before we give a full description of ClearBox in Section 3, we present in this section our system and security models.

在第 3 节给出 ClearBox 的完整描述之前，我们在这一节介绍我们的系统和安全模型。

2.1 System Model

2.1 系统模型

ClearBox comprises a number of clients C1, C2,...that are in-terested in storing their files at a storage provider S such that each client learns the level of deduplication undergone by his files, that is the number of clients that outsourced the same file.Since exist-ing storage providers do not report the deduplication patterns of the stored data, a straightforward approach would be to rely on a decen-tralized scheme whereby users coordinate their file uploads prior to storing their data onto the cloud;such a decentralized scheme, however, requires interaction among users, and is unlikely to scale as the number of users storing the same data increases [41].This is exactly why ClearBox makes use of an additional gateway G, which interfaces between the users and the cloud, and orchestrates data deduplication prior to storing the data on the cloud.Note that G is a logically centralized entity, and can be easily instantiated using distributed servers.We argue that our model is generic and captures a number of practical deployment scenarios.For instance, G could be an independent service which offers cheaper cloud stor-age, by performing data deduplication over existing public clouds;alternatively, G could be a service offered by S itself in order to offer differentiation to existing cloud storage services, etc.

ClearBox 由 C1、C2，...他们有兴趣将其文件存储在存储提供商处，以便每个客户端了解其文件经历的重复数据消除级别，即外包相同文件的客户端数量。由于现有的存储提供商不报告存储数据的重复数据消除模式，因此一种简单的方法是依赖于一种集中化的方案，即用户在将数据存储到云中之前协调他们的文件上传；然而，这种分散的方案需要用户之间的交互，并且不太可能随着存储相同数据的用户数量的增加而扩展[41]。这就是为什么 ClearBox 使用了一个额外的网关 G，它连接用户和云，并在将数据存储到云上之前编排重复数据删除。请注意，G 是一个逻辑上集中的实体，可以很容易地使用分布式服务器进行实例化。我们认为我们的模型是通用的，并且捕捉了许多实际的部署场景。例如，通过在现有公共云上执行重复数据消除，G 可以是一个独立的服务，提供更便宜的云存储；或者，G 可以是 S 自己提供的服务，以便为现有的云存储服务提供差异化，等等。

In our scheme, we assume that G owns an account hosted by S, and orchestrates cross-user file-based deduplication.By doing so, G can provide its users with full transparency on the storage savings due to deduplication exhibited by their data.For instance, G can offer price reductions for customers whose data is highly deduplicated (e.g., 50% discount if the file is deduplicated among at least n entities).Alternatively, G could fairly distribute storage costs amongst users who are storing deduplicated files;for exam-ple, assume that there are n clients all storing the same file f, then each client can effectively be charged a fraction of n of the cost of storing f. Notice that these reductions do not threaten the prof-itability of G (nor S) which can still comfortably profit from the various offered services (e.g., resilience to failures, mobile access, data synch across devices).On the contrary, we argue that offer-ing the option of sharing storage costs with other clients storing the same files may provide a clear differentiator compared to other competitors.Nevertheless, providers are clearly not keen on shar-ing parts of their profits with the users and may not correctly report the cost reductions due to data deduplication.The challenge here Although block-based deduplication can lower storage consump-tion to as little as 32% of its original requirements, we point out that nearly three quarters of the improvement observed could be captured through whole-file deduplication [36].

在我们的方案中，我们假设 G 拥有一个由 S 托管的帐户，并协调基于文件的跨用户重复数据消除。通过这样做，G 可以向其用户提供完全透明的存储节省，这是由于其数据表现出的重复数据消除。例如，G 可以为数据经过高度重复数据消除的客户提供价格优惠(例如，如果文件在至少 n 个实体中进行了重复数据消除，则可享受 50%的折扣)。或者，G 可以在存储重复数据消除文件的用户之间公平分配存储成本；例如，假设有 n 个客户机都存储相同的文件 f，那么每个客户机可以有效地被收取存储 f 的成本的一小部分 n。请注意，这些减少不会威胁到 G (nor S)的可实现性，G(nor S)仍然可以从提供的各种服务(例如，故障恢复能力、移动访问、跨设备的数据同步)中轻松获利。相反，我们认为，提供与存储相同文件的其他客户端共享存储成本的选项，与其他竞争对手相比，可能会提供明显的优势。然而，提供商显然不热

衷于与用户分享部分利润，并且可能无法正确报告重复数据消除带来的成本降低。这里的挑战尽管基于数据块的重复数据消除可以将存储消耗降低到最初需求的 32%，但我们指出，观察到的改进中有近四分之三可以通过全文件重复数据消除获得[36]。

887

887

therefore lies in transparently and efficiently attesting data stor-age costs (i.e., including deduplication savings) across users in the presence of a storage provider which might not correctly report the deduplication (and access) patterns of the data that it is storing.

因此，关键在于在存储提供商可能无法正确报告其存储的数据的重复数据消除(和访问)模式的情况下，透明、高效地证明用户之间的数据存储成本(即，包括重复数据消除节省)。

Conforming with the operation of existing storage providers, we assume that time is divided into epochs Ei of equal length, e.g., 12 hours [1].Clients receive from G a list of their files and deduplica-tion patterns at the end of every epoch.The deduplication pattern of a given file refers to the number of users that store the same deduplicated file in the cloud.

与现有存储提供商的操作一致，我们假设时间被分成等长的时期 Ei，例如 12 小时[1]。客户在每个时期结束时都会从 G 那里收到一份文件和重复数据消除模式的列表。给定文件的重复数据消除模式是指云中存储相同重复数据消除文件的用户数量。

Similar to existing storage providers, ClearBox supports the fol-lowing operations: Put, Get, Delete.In addition, ClearBox sup-ports two additional protocols, Attest and Verify, which are used to prove/verify the deduplication patterns undergone by files.We start by describing these protocols.Here, the expression

与现有的存储提供商类似，ClearBox 支持以下操作:放、获取、删除。此外，ClearBox 支持两个附加协议，即证明和验证，用于证明/验证文件经历的重复数据消除模式。我们从描述这些协议开始。在这里，表达

Π:[P1 : in1;..., Pn : inn] → [P1 : out1;..., Pn : outn]

【P1:1；...，Pn:inn]→[P1:out 1；...，Pn : outn]

denotes the event that a set of parties P1,...,Pn run an interactive protocol Π where ini and outi denote the input and output of Pi, respectively.

表示一组当事人 P1，...，Pn 运行一个交互式协议 π，其中 ini 和 outi 分别表示 Pi 的输入和输出。

The Put Protocol.

卖出协议。

The Put protocol is executed between the gateway and a client C who aims to upload a file f. Initially, the client derives a key kFID from the file which he uses to encrypt f to f which is even-tually uploaded to S via G. Next, both parties derive a file ID FID that will serve as a practically unique handle to f. The gateway G maintains internally a set F which contains pairs (FID, CFID) where CFID is the set of clients that are registered to the file refer-enced by FID.If no client is registered to a file with file ID FID, it holds that F does not contain an element of the form (FID, *).Given this, the gateway checks if any users are registered to this file already, i.e., if (FID, CFID) ∈ F. If so, it inserts C into CFID .Otherwise, a new set CFID = {C} is created and (FID, CFID) in-serted into F. Moreover, the client gets a verification tag τ which later allows to validate the proofs generated by G.

Put 协议在网关和旨在上传文件 F 的客户端 C 之间执行。最初，客户端从文件中导出密钥 kFID，他使用该密钥将 F 加密为 F，该密钥通过 G 最终上传到 S。接下来，双方导出文件 ID FID，该文件 ID 将作为 F 的实际唯一句柄。网关 G 在内部维护一个包含对(FID，CFID)的集合 F，其中 CFID 是注册到 FID 所引用的文件的客户端集合。如果没有客户端注册到文件标识为 FID 的文件，它会认为 F 不包含表单元素(FID，*)。考虑到这一点，网关检查是否有任何用户已经注册到这个文件，即如果(FID，CFID) ∈ F .如果有，它将 C 插入到 CFID。否则，将创建一个新的集合 CFID = {C}，并将(FID，CFID)插入到 f 中。此外，客户端将获得一个验证标记 τ，该标记允许验证 g 生成的证明

Put : [C : f ;G : F ;S : ⊥] −→

放:[C:f；g:F；s:⊥]→

[C : FID, kFID , τ ;G : FID , F ;S : f]

[C : FID，kFID，τ；G : FID，F；S : f]

When we need to specify the verification tag of a specific client C, we write τC .

当我们需要指定特定客户端 C 的验证标签时，我们写 τC。

The Get Protocol.

获取协议。

When a client C wants to download a file f, it initiates this proto-col with G and sends the corresponding file ID FID.The gateway first checks if F contains an entry (FID, CFID).In the positive case, it further verifies if C ∈ CFID .If this verification passes, actions are taken such that eventually the client can download the encrypted file f from S and decrypt it to f with kFID .Observe that we do not specify additional tokens to authenticate the client as we assume the existence of authenticated channels between the clients and G.

当客户端 C 想下载一个文件 f 时，它用 G 启动这个原型，并发送相应的文件标识 FID。网关首先检查 F 是否包含条目(FID，CFID)。在正的情况下，它进一步验证 C ∈ CFID。如果验证通过，将采取措施，最终客户端可以从 S 下载加密文件 f，并用 kFID 将其解密为 f。请注意，我们没有指定额外的令牌来验证客户端，因为我们假设客户端和 g 之间存在经过验证的通道。

Get : [C : FID, kFID ;G : F ;S : f] −→

get:【C:FID，kFIDg:F；s:f]→

[C : f ;G : ⊥ ;S : ⊥]

[C:f；g:⊥；学生:⊥]

The Delete Protocol.

删除协议。

This protocol allows a client to delete a file or, more precisely, to cancel his registration.To this end, the client sends the file ID FID to the gateway who checks if (FID, CFID) ∈ F for some set CFID and if C ∈ CFID .If this is not the case, the request is sim-ply ignored and C receives f = ⊥.Otherwise CFID is updated to CFID \ {C} at the beginning of the next epoch.If CFID becomes

该协议允许客户端删除文件，或者更准确地说，取消注册。为此，客户端将文件 ID FID 发送到网关，网关检查是否(FID，CFID)←F 为某个集合 CFID，是否 C ∈ CFID。如果不是这种情况，请求被简单地忽略，c 接收 f = ⊥.否则，CFID 将在下一个纪元开始时更新为 CFID。如果 CFID 成为

empty by this action, this means that no user is any longer regis-tered to this file.Hence, G can request to S to delete the file.

此操作为空，这意味着没有用户再注册此文件。因此，G 可以请求 S 删除该文件。

Delete : [

删除:[

C : FID ;G : F ;S : f] −→

c:FID；g:F；s:f]→

[C : ⊥ ;G : F ;S : ⊥]

[c:⊥；g:F；学生:⊥]

The Attest Protocol.

证明协议。

The purpose of the Attest procedure, which is executed by the gateway only, is twofold.On the one hand, it generates a proof of cardinality for a given file ID FID and an epoch E that attests an upper bound for |CFID |—the number of clients registered to this file within this epoch.On the other hand, it also includes a proof of membership for a given client C with respect to CFID .Formally, we have:

仅由网关执行的证明过程有两个目的。一方面，它为给定的文件标识 FID 和纪元 E 生成基数证明，证明了|CFID |的上限，即在此纪元内注册到此文件的客户端数量。另一方面，它还包括给定客户 C 关于 CFID 的成员资格证明。形式上，我们有:

A ← Attest(FID,E, CFID , C).

A ←证明(FID、E、CFID、C)。

The proof A, i.e., the output of Attest, contains a claim on an upper bound of |CFID |, a compact digest for CFID , and possibly additional information, so that a client C ∈ CFID can use the subsequent Verify to get convinced of the upper bound of the set CFID and its own membership.As described in Section 2.2, an upper bound for |CFID | is sufficient and necessary in our model.Namely, G does not have incentives to report a larger value of |CFID | since this results in G under-charging clients and hence a reduction of the profit.Therefore, to increase its profits, G's interest is to claim the smallest possible upper bound at the end of each epoch.

证明 A，即证明的输出，包含对|CFID |的上界的声明、CFID 的紧凑摘要以及可能的附加信息，以便客户端 C ∈ CFID 可以使用后续的验证来确信集合 CFID 的上界及其自身的成员资格。如第 2.2 节所述，|CFID |的上限在我们的模型中是充分必要的。也就是说，G 没有动力报告更大的价值|CFID |因为这导致 G 对客户收费过低，因此利润减少。因此，为了增加利润，G 的兴趣是在每个纪元结束时要求最小可能的上限。

The Verify Protocol.

验证协议。

In ClearBox, customers can verify (using the Verify protocol) the proof generated by the Attest protocol to confirm that they are part of the set of file users, and to verify the upper bound on the total number of file users.The Verify algorithm is executed by a client, and uses the verification tag which has been generated during the Put procedure.

在 ClearBox 中，客户可以验证(使用验证协议)由证明协议生成的证明，以确认他们是文件用户集的一部分，并验证文件用户总数的上限。验证算法由客户端执行，并使用在 Put 过程中生成的验证标记。

It outputs either accept or reject to indicate whether the proof is accepted or not.

它输出接受或拒绝，以指示证据是否被接受。

accept|reject ← Verify(FID,E, A, τ).

接受|拒绝←验证(FID，E，A，τ)。

2.2 Security Model

2.2 安全模型

In the sequel, we assume that the communication between a client and the gateway is authenticated to provide non-repudiation and, in the case of need, encrypted.As already stated, we assume that time is divided into a sequence of epochs E1, E2,..., where E ≤ Emeans that E either took place before epoch Eor that both refer to the same epoch.If not mentioned otherwise, we will always refer to epochs that happened in the past.Moreover, we assume that all parties are synchronized.That is, at each point in time, all parties share the same view on the current epoch (e.g., its index number if epochs are represented by an increasing counter).

在后续文章中，我们假设客户端和网关之间的通信经过身份验证，以提供不可否认性，并且在需要时进行加密。如前所述，我们假设时间被分成一系列的时期 E1，E2，...，其中 E≤E 表示 E 要么发生在纪元之前，要么都指同一个纪元。如果没有另外提及，我们将总是提及过去发生的时代。此外，我们假设各方都是同步的。也就是说，在每个时间点，所有各方对当前纪元都有相同的看法(例如，如果纪元由递增的计数器表示，则其索引号)。

In each epoch, several protocol runs may be executed between the gateway G and a client.A protocol run is represented by a quadruple pi = (C, prot,(in),(out)) where C is a client, prot denotes one of the protocols Put, Get, Delete, in denotes the inputs of C, and out its outputs.For any past epoch E, we denote by PE all protocol runs that occurred within this epoch.Here, we restrict ourselves to full protocol runs, i.e., which have not been aborted.We assume that these are uniquely ordered within the epoch.That is for PE = {p1,...,p

在每个时期，在网关 G 和客户端之间可以执行几个协议运行。协议运行由一个四元组 pi = (C，prot，(in)，(out))表示，其中 C 是客户端，prot 表示协议 Put，Get，Delete 之一，in 表示 C 的输入，out 表示其输出。对于任何过去的纪元 E，我们用 PE 表示在这个纪元内发生的所有协议运行。在这里，我们将自己限制为完整的协议运行，即没有中止的运行。我们假设这些在纪元内是唯一有序的。也就是说对于 PE = {p1，...，p

-} where

哪里

denotes the total number of protocol runs within this epoch, it holds for any p, p∈ PE with p = pthat either p<p(when p took place before p) or p< p (when phappened first).We extend this order to the set of all protocol runs in a straightforward way.More precisely, for any two epochs E = Ewith E<Eit holds that p<pfor all p ∈ PE

表示该时期内协议运行的总数，它适用于任何 p，p∈p = p 的 PE，p<p(当 p 在 p 之前发生时)或 p< p(当 p 首先发生时)。我们以简单的方式将这个顺序扩展到所有协议运行的集合。更准确地说，对于任何两个时代，E = Ewith E<Eit 认为 p<pfor 所有 p ∈ PE

888

888

and all p∈ PE.Finally, we denote by P≤E =

和所有 p∈ PE。最后，我们用 P≤E =表示

- E

-英

-≤E PEall protocol runs that took place up to epoch E inclusive and define P<E analogously.

-≤E PEall 协议运行发生在纪元 E(含)之前，并以类似方式定义 P<E。

We say that the scheme provides correct attestation if under the conditions of Ereg(E, C, f, FID, τ) and |CFID (E)| ≤ bd it holds that:

如果在 Ereg(E，C，f，FID，τ)和|CFID (E)| ≤ bd 的条件下，认为:

P r [accept ← Verify(FID,E, A, τ)|]=1.

p r[接受←验证(FID，E，A，τ)|]=1。

(4)

(4)

DEFINITION 1 (FILE REGISTRATION).

定义 1(档案登记)。

We say that a client C is registered to a file ID FID at some epoch E if there exists a p = (C, Put,(f),(FID, kFID , τ)) ∈ P≤E for which one of the following two conditions hold:

我们说，如果存在满足以下两个条件之一的 p = (C，Put，(f)，(FID，kFID，τ)) ∈ P≤E，则客户端 C 在某个时间 E 注册到文件 ID FID:

1.p ∈ PE

1.p ∈ PE

2.p ∈ P<E and for all p∈ P<E with p<pit holds that

2.p ∈ P<E，对于所有 p∈ P<E，p<pit 认为

p = (C, Delete,(FID),(⊥)).

Delete,(FID),(⊥)).

We denote by CFID (E) the set of all clients that are registered to FID at epoch E. If the considered epoch is clear from the context, we simply write CFID .Finally, we denote by Ereg(E, C, f, FID, τ) the event that C is registered to FID within epoch E where f is the file specified in the protocol run p mentioned above (that is the file used in the last upload).Obviously, it holds that C ∈ CFID (E) if and only if Ereg(E, C, f, FID, τ) holds for some file f.

我们用 CFID (E)表示在纪元 E 注册到 FID 的所有客户的集合。如果考虑的纪元从上下文中是清楚的，我们简单地写 CFID。最后，我们用 Ereg(E，C，f，FID，τ)表示 C 在纪元 E 内注册到 FID 的事件，其中 f 是上述协议运行 p 中指定的文件(即上次上传中使用的文件)。显然，它认为 C ∈ CFID (E)当且仅当 Ereg(E，C，f，FID，τ)对某些文件 f 成立

Soundness

健康

We assume that each party (client, gateway, service provider) can potentially misbehave but aim for different attack goals.Conse-quently, soundness requires that security is achieved against all these attackers.In what follows, we motivate each attacker type and define the corresponding security goals.

我们假设每一方(客户端、网关、服务提供商)都可能行为不端，但目标不同。因此，健全性要求针对所有这些攻击者实现安全性。在接下来的内容中，我们将激励每个攻击者类型，并定义相应的安全目标。

Malicious Clients: In principle, we assume that a client may be arbitrarily malicious with various aims: disrupting the service, re-pudiate actions, and gain illegitimate access to files.The first two can be thwarted with standard mechanisms like authenticated chan-nels.Thus, we focus on the third only: a user must only be able to access a file f via Get if he had uploaded it before to G and if he is still registered to it.

恶意客户端:原则上，我们假设客户端可能是任意恶意的，有各种目的:破坏服务、阻止操作和非法访问文件。前两者可以用标准机制来阻止，比如认证通道。因此，我们只关注第三个:如果用户之前已经将文件 f 上传到 G，并且他仍然注册到 G，那么他必须只能通过 Get 访问文件 f。

The first condition in Def.1 means that if C uploaded the file within epoch E, he is registered to the file, no matter if he requests to delete it later in the same epoch.The second condition means that if the upload took place in a previous epoch, C must not have sub-sequently asked to delete it in an older epoch.Observe that also here, if C asks to delete it within epoch E, he is still registered to FID within this epoch.Likewise, we allow the client to download the file any time within an epoch if he is registered to the file within this epoch.

Def.1 中的第一个条件意味着，如果 C 在 E 纪元内上传了文件，他将被注册到该文件，无论他是否在同一纪元的稍后请求删除该文件。第二个条件意味着，如果上传发生在前一个时期，C 不能在更早的时期被要求删除它。还要注意，在这里，如果 C 要求在纪元 E 内删除它，他仍然在这个纪元内注册到 FID。同样，如果客户在一个时期内注册了文件，我们允许他在这个时期内的任何时间下载文件。

We are now ready to formally define correctness and soundness.

我们现在准备正式定义正确性和可靠性。

DEFINITION 4 (SECURE FILE ACCESS).We say that the sche-me provides ε-secure file access if it holds for any client C, for any epoch E, and any file f that for any (C, Get,(FID, kFID),(f)) ∈ PE while Ereg(E, C, f, FID, τ) does not hold, then

定义 4(安全文件访问)。我们说 sche-me 提供 ε-安全的文件访问，如果它适用于任何客户端 C、任何纪元 E 和任何文件 f，而 Ereg(E、C、f、FID、τ)则不适用

P r

公关

f= ⊥ ≥ 1 − ε.

f =⊥≥1ε。

(5)

(5)

Rational Gateway: We assume that both the gateway and the ser-vice provider are rational entities, e.g., see [43] for a similar as-sumption.By rational, we mean that the gateway and the storage provider will only deviate from the protocol if such a strategy in-creases their profit in the system.Notice that gateway has access to all the information witnessed by the service provider (e.g., access to the stored files).This implies that any attack by a rational service provider may equally be executed by the gateway.Likewise, a ra-tional gateway could not mount stronger attacks by colluding with the service provider.Hence, we restrict our analysis in the sequel to the security of our scheme given a rational gateway.

Rational Gateway:我们假设网关和服务提供者都是 Rational 实体，例如，类似的假设见[43]。理性地说，我们的意思是网关和存储提供商只有在这样的策略增加了他们在系统中的利润时才会偏离协议。请注意，网关可以访问服务提供商见证的所有信息(例如，访问存储的文件)。这意味着任何由 rational 服务提供者发起的攻击都可能由网关执行。同样，国家网关不能通过与服务提供商勾结来发动更强的攻击。因此，在给定一个合理的网关的情况下，我们将我们的分析限制在我们方案的安全性上。

Recall that the gateway performs two types of interactions: at-testing the number of registered clients and handling clients' files.Consequently, we see two different types of strategies that may al-low the gateway to increase its advantage in the system: (i) over-charging clients and (ii) illegitimately extracting file content.

回想一下，网关执行两种类型的交互:at 测试注册客户端的数量和处理客户端的文件。因此，我们看到了两种不同类型的策略，它们可能会降低网关以增加其在系统中的优势:(I)向客户端过度收费和(ii)非法提取文件内容。

With respect to the first strategy, if a rational gateway manages to convince clients of a smaller level of deduplication, the gateway can charge higher prices.On the other hand, with respect to the sec-ond strategy, the gateway may try to derive information about the contents of the uploaded files;notice that acquiring information about users' data can be economically beneficial for the gateway since the stored data can be misused, e.g. sold.Consequently, two security goals need to be ensured: secure attestation and data con-fidentiality, that we formalize next.

关于第一个策略，如果一个 rational gateway 能够说服客户端使用更低级别的重复数据消除，那么这个 gateway 可以收取更高的价格。另一方面，关于第二策略，网关可以尝试导出关于上传文件内容的信息；请注意，获取关于用户数据的信息对于网关来说在经济上是有益的，因为存储的数据可能被滥用，例如被出售。因此，需要确保两个安全目标:安全证明和数据可信度，这是我们接下来要形式化的。

Correctness

正确性

For correctness, two requirements arise.First, a user who uploaded a file with Put, must be able to obtain it with Get for those epochs during which he is registered to this file.As files are identified by their file ID, we address the file ID generation process first.Namely, we say that the file ID generation process creates ε-unique file IDs if it holds for any two protocol runs(C, Put,(f),(FID, kFID)) and (C, Put,(f),(FID, kFID)) that

对于正确性，有两个要求。首先，一个用 Put 上传文件的用户，必须能够在他注册到这个文件的那些时期用 Get 获得它。由于文件是由它们的文件标识来标识的，我们首先处理文件标识生成过程。也就是说，如果文件标识生成过程适用于任意两个协议运行(C，Put，(f)，(FID，kFID))和(C，Put，(f)，(FID，kFID))，则它会创建 ε 唯一的文件标识

P r

公关

FID = FID|f = f = 1, (1)

FID = FID|f = f = 1，(1)

P r

公关

FID = FID|f = f ≤ ε.(2)

FID = FID|f = f ≤ ε。(2)

The first condition means that the same file leads always to the same file ID while the probability that different files result into the same file ID is at most ε.This allows us to define correct access for our scheme:

第一个条件意味着相同的文件总是导致相同的文件标识，而不同的文件导致相同的文件标识的概率最多为 ε。这允许我们为我们的方案定义正确的访问权限:

DEFINITION 2 (CORRECT ACCESS).Assume that the file ID generation process is ε-unique.We say that the scheme provides correct access if it holds for any client C, for any epoch E, and any file f that if the event Ereg(E, C, f, FID) holds and if there exists a protocol run (C, Get,(FID, kFID),(f)) ∈ PE, then:

定义 2(正确访问)。假设文件标识生成过程是 ε 唯一的。我们说，如果该方案为任何客户端 C、任何纪元 E 和任何文件 f 提供正确的访问，如果事件 Ereg(E、C、f、FID)成立，并且如果存在协议运行 (C、Get、(FID、kFID)、(f)) ∈ PE，则:

P r

公关

f= f ≥ 1 − ε.

f = f≥1ε。

(3)

(3)

The reason that one cannot require the probability to be equal to 1 is that with some probability ε, two different files may get the same file ID.In these cases, correctness cannot be guaranteed anymore.

之所以不能要求概率等于 1，是因为在某种概率 ε 下，两个不同的文件可能会得到相同的文件 ID。在这些情况下，正确性再也无法保证了。

DEFINITION 3 (CORRECT ATTESTATION).Let FID be an ar-bitrary file ID and E be an arbitrary epoch.Moreover, let A be a proof generated by the gateway for a file FID and epoch E. That is, A ←

Attest(FID,E, CFID).Moreover, let bd denote the upper bound for |CFID (E)| claimed in A and let C be an arbitrary client.

定义 3(正确证明)。假设 FID 是一个随机文件标识，E 是一个任意的纪元。此外，假设 A 是网关为文件 FID 和纪元 E 生成的证明，即 A ←证明(FID，E，CFID)。此外，让 bd 表示在 A 中声明的|CFID (E)|的上限，让 C 是任意客户。

DEFINITION 5 (SECURE ATTESTATION).Let FID be an ar-bitrary file ID and E be an arbitrary epoch.Moreover, let A be a proof generated by the gateway for a file FID, an epoch E,and a client C, that is A ← Attest(FID,E, CFID , C).Moreover, let bd denote the upper bound for |CFID (E)| claimed in A. We say that the scheme provides ε-secure attestation if Ereg(E, C, f, FID, τ) or |CFID (E)| > bd implies that

定义 5(安全证明)。假设 FID 是一个随机文件标识，E 是一个任意的纪元。此外，假设 A 是网关为文件 FID、纪元 E 和客户端 C 生成的证明，即 A ←证明(FID，E，CFID，C)。此外，让 bd 表示在 a 中声明的|CFID (E)|的上界。我们说，如果 Ereg(E，C，f，FID，τ)或|CFID (E)| > bd 暗示，该方案提供 ε-安全证明

P r [accept ← Verify(FID,E, A, τ)|] ≤ ε.We assume the gateway to be curious in this respect.

p r[接受←验证(FID，E，A，τ)|] ≤ ε。我们假设网关在这方面很好奇。

(6)

(6)

889

Observe that it is not in the interest of a rational gateway in our model to report a larger value for |CFID (E)| as this would allow the clients to demand further cost reductions.

请注意，在我们的模型中，报告|CFID (E)|的较大值不符合 rational gateway 的利益，因为这将允许客户要求进一步降低成本。

With respect to the confidentiality of the uploaded data, notice that standard semantically secure encryption schemes cannot be used in our context since they effectively prevent the deduplica-tion of data [14, 25, 42].Schemes that are suitable for deduplica-tion produce the same ciphertext for multiple encryptions of the same message and are referred to as message-locked encryption (MLE) [13, 15].The achievable security in our case is therefore that of MLE schemes.

关于上传数据的机密性，请注意标准语义安全加密方案不能用于我们的上下文，因为它们有效地防止了数据的重复[14，25，42]。适用于重复数据删除的方案为同一消息的多次加密产生相同的密文，被称为消息锁定加密(MLE) [13，15]。因此，在我们的案例中，可实现的安全性是 MLE 方案的安全性。

To describe the security of MLE schemes, we adapt in what fol-lows the security notion introduced in [15] which guarantees pri-vacy under chosen distribution attacks.

为了描述最大似然估计方案的安全性，我们采用了[15]中引入的安全性概念，它保证了在选择分布攻击下的安全性。

An MLE scheme consists of a tuple of algorithms (setup, keygen, enc,dec, tag), where setup generates (public) parameters P, the key generation keygen generates a key k given P and a message f;finally, enc and dec are the algorithms to encrypt and decrypt f with key k. The tag generation algorithm tag creates a tag for a ciphertext.This is covered in our model by the unique file identifier FID and does not play a role in the privacy notion.

MLE 方案由一组算法(setup、keygen、enc、dec、tag)组成，其中 setup 生成(公共)参数 P，key generation 生成给定 P 的密钥 k 和消息 f；最后，enc 和 dec 是用密钥 k 加密和解密 f 的算法。标签生成算法标签为密文创建一个标签。在我们的模型中，这是由唯一文件标识符 FID 涵盖的，在隐私概念中不发挥作用。

A message space is given by an algorithm M which samples messages $M \in \{0, 1\}$according to a distribution and may provide additional context information.The message sampling must be un-predictable, i.e., the probability to predict the output of M is neg-ligible given context information.The security of an MLE scheme is defined by the following experiment PRV\$-CDAA MLE,M, an un-predictable sampling algorithm M and an adversary A [15]:

消息空间由算法 M 给出，该算法根据分布对消息 M∞{ 0，1 }进行采样，并且可以提供附加的上下文信息。消息采样必须是不可预测的，即在给定上下文信息的情况下，预测 M 的输出的概率是不可忽略的。最大似然估计方案的安全性由以下实验定义:PRV-CDAA 最大似然估计，M，不可预测抽样算法 M 和对手 A [15]:

1.The environment randomly generates a parameter set P and a bit $b \leftarrow \{0, 1\}$.

1.环境随机生成一个参数集 P 和一个位 $b \leftarrow \{0，1\}$。

2.The environment samples a set of messages M and context information Z: $(M,Z) \leftarrow M$. As the same message would result in the same encryption, the restriction here is that all messages are distinct.

2.环境采样一组消息 M 和上下文信息 Z: $(M，Z) \leftarrow M$。由于相同的消息将导致相同的加密，这里的限制是所有消息都是不同的。

3.For all messages M[i] in M, the environment encrypts mes-sage M[i] with the MLE scheme:

3.对于 M 中的所有消息 M[i]，环境使用 MLE 方案加密消息 M[i]:

$C0[i] \leftarrow enckeygen(M[i])(M[i])$

$C0[i] \leftarrow enckeygen(M[i])(M[i])$

and generates a random string of the same length:

并生成相同长度的随机字符串:

$C1[i] \leftarrow \{0, 1\}|C[i]|$

C1[I]；{ 0，1}|C[i]|

.

。

4.The adversary obtains the set Cb and outputs a guess for b: b ← A(P, Cb, Z)

4.对手获得集合 Cb 并输出对 b 的猜测:b ← A(P，Cb，Z)

5.The adversary wins if b equals b in this case the experiment returns 1, otherwise 0.

5.如果 b 等于 b，对手就赢了。在这种情况下，实验返回 1，否则返回 0。

DEFINITION 6 (DATA CONFIDENTIALITY).We say that the MLE scheme for message sampling algorithm M is secure, if the adversary's advantage in winning the PRV$-CDAA MLE,M experi-ment is negligible, i.e.

定义 6(数据保密)。我们说消息抽样算法 M 的极大似然估计方案是安全的，如果对手在赢得 PRV-CDAA 极大似然估计中的优势可以忽略不计，即

P r ← PRV$-CDA A MLE, M − ≤ negl(κ) ,

P r ← PRV$-CDA A MLE，M≤negl(κ)，

for a security parameter κ.

对于安全参数 κ。

Notice that it is integral for both G and S to know the file sizes and the download patterns of files in order to perform correct ac-counting.Therefore, hiding this information cannot be part of our goals.

请注意，为了执行正确的计数，G 和 S 都必须知道文件大小和文件的下载模式。因此，隐藏这些信息不能成为我们目标的一部分。

Current cloud services maintain a detailed log containing all the activities per account.

当前的云服务维护包含每个帐户所有活动的详细日志。

Command	Description
createBucket(B) PUT(B, FID) GET(B, FID) DELETE(B, FID) generateURL(COMMAND, t)	Creates a bucket B Upload a file FID to B Download a file FID from B Delete a file FID from B Generate a URL expiring at time t supporting PUT, GET, DELETE.

命令	描述

创建桶放获取删除删除 生成器 1(命令，测试)	创建一个桶 B 上传一个文件 FID 到 B 从 B 下载一个文件 FID 从 B 删除一个文件 FID 生成一个在支持 PUT，GET，Delete 的时间 t 过期的 URL。

Table 1: Sample API exposed by Amazon S3 and Google Cloud Storage.COMMAND refers to an HTTP command such as PUT(B, FID).

表 1:亚马逊 S3 和谷歌云存储公开的示例 API。COMMAND 指的是一个 HTTP 命令，比如 PUT(B，FID)。

2.3 Design Goals

2.3 设计目标

In addition to the security goals stated above, our proposed solu-tion should satisfy the following functional requirements.ClearBox should work within the APIs provided by current service providers, without deteriorating the performance witnessed by users when compared to standard solutions where users directly interface with S. Similar to existing cloud storage providers such as Amazon S3 and Google Cloud Storage, we assume that S exposes to its clients a standard interface consisting of a handful of basic operations, such as storing a file, retrieving a file, deleting a file, generating a signed URL for sending HTTP commands for storage/retrieval, etc.(cf. Table 1).Where appropriate, we also discuss the case where S is a commodity cloud service provider and exposes a simpler interface, e.g., that does not allow storing a file using a URL.

除了上述安全目标，我们提出的解决方案还应满足以下功能要求。与用户直接与 S 接口的标准解决方案相比，ClearBox 应该在当前服务提供商提供的 API 内工作，而不会降低用户见证的性能。类似于现有的云存储提供商，如亚马逊 S3 和谷歌云存储，我们假设 S 向其客户端公开了一个标准接口，该接口由少量基本操作组成，如存储文件、检索文件、删除文件、生成用于发送存储/检索 HTTP 命令的签名 URL 等。(参见表 1)。在适当的情况下，我们还讨论了 S 是商品云服务提供商的情况，并公开了一个更简单的界面，例如，不允许使用 URL 存储文件。

Moreover, our solution should scale with the number of users, the file size, and the number of uploaded files, and should incur tolerable overhead on the users when verifying the deduplication patterns of their files at the end of every epoch.

此外，我们的解决方案应该随着用户数量、文件大小和上传文件的数量而扩展，并且在每个时期结束时验证用户文件的重复数据消除模式时，应该会给用户带来可承受的开销。

3.ClearBox

3.白盒

In this section, we present our solution, and analyze its security according to the model outlined in Section 2.

在这一节中，我们介绍我们的解决方案，并根据第 2 节中概述的模型分析其安全性。

3.1 Overview

3.1 概述

ClearBox ensures a transparent attestation of the storage con-sumption of users whose data is effectively being deduplicated— without compromising the confidentiality of the stored data.

ClearBox 确保对其数据正在进行有效重复数据消除的用户的存储消费进行透明证明，而不会损害存储数据的机密性。

To attest the deduplication patterns to its customers, one naive solution would be for the gateway to publish the list of all clients associated to each deduplicated file (e.g., on a public bulletin board) such that each client could first check if (i) he is a member of the list and (ii) if the size of the list corresponds to the price reduction offered by the gateway for storing this file.Besides the fact that this solution does not scale, it is likewise within the interest of G not to publish the entire list of its clients and their files;for exam-ple, competitors could otherwise learn information on the service offered by G (e.g., total turnover).

为了向其客户证明重复数据消除模式，一个简单的解决方案是网关发布与每个重复数据消除文件相关联的所有客户端的列表(例如，在公共公告板上)，以便每个客户端可以首先检查(I)他是否是该列表的成员，以及(ii)该列表的大小是否对应于网关为存储该文件而提供的价格降低。除了该解决方案不可扩展的事实之外，不公布其客户及其文件的整个列表同样符合 G 的利益；例如，竞争对手可以通过其他方式了解通用汽车提供的服务信息(例如，总营业额)。

To remedy this, ClearBox employs a novel Merkle-tree based cryptographic accumulator which is maintained by the gateway to efficiently accumulate the IDs of the users registered to the same file within the same time epoch.Our construct ensures that each user can check that his ID is correctly accumulated at the end of ev-ery epoch.Additionally, our accumulator encodes an upper bound on the number of accumulated values, thus enabling any legitimate client associated to the accumulator to verify (in logarithmic time with respect to the number of clients that uploaded the same file) this bound.

为了解决这个问题，ClearBox 采用了一种新颖的基于 Merkle 树的密码累加器，该累加器由网关维护，以有效地累加在同一时间周期内注册到同一文件的用户的标识。我们的构造确保每个用户都能在每个时期结束时检查他的 ID 是否正确累积。此外，我们的累加器对累加值的数量进行上限编码，从而使任何与累加器相关联的合法客户端都能够验证(相对于上传同一文件的客户端数量，以对数时间表示)该界限。

Clearly, a solution that requires the gateway to publish details about all the accumulators for all the stored files does not scale with the number of files stored in the cloud.This is why ClearBox relies on a probabilistic algorithm which selectively reveals details about a number of file accumulators in each epoch.However, if the gate-

显然，要求网关发布所有存储文件的所有累加器的详细信息的解决方案不会随着云中存储的文件数量而扩展。这就是为什么 ClearBox 依赖于一种概率算法，这种算法有选择地揭示每个时期的文件累加器数量的细节。但是，如果大门-

890

way could select which file accumulators to publish, then G could easily cheat by only creating correct accumulators for the selected files, while misreporting the deduplication patterns of the remain-ing files.In ClearBox, the choice of which accumulators are pub-lished in each epoch is seeded by an external source of randomness which cannot be predicted but can be verified by any entity.We show how such a secure source of randomness can be efficiently instantiated using Bitcoin.This enables any client to validate that the sampling has been done correctly, and as such that he is acquir-ing the promised price reductions—without giving any advantage for G to misbehave.

方法可以选择要发布的文件累加器，然后 G 可以通过只为所选文件创建正确的累加器，而错误地报告剩余文件的重复数据消除模式，来轻松作弊。在 ClearBox 中，选择在每个时期发布哪些累加器是由外部随机源播种的，这些随机源无法预测，但可以由任何实体验证。我们展示了如何使用比特币有效地实例化这样一个安全的随机性来源。这使得任何客户都能够验证取样是否正确，以及他是否获得了承诺的降价——而不会给 G 带来任何不当行为的好处。

ClearBox enforces fine-grained access control on shared files by leveraging self-expiring URLs when accessing content.Namely, whenever a user wishes to access a given resource, the gateway generates a URL for that resource on the fly, which expires after a short period of time.As shown in Table 1, existing cloud APIs support the dynamic generation of expiring URLs.By doing so, ClearBox does not only ensure that G can restrict access to the data stored on the cloud, but also enables G to keep track of the access patterns of its users (e.g., to be used in billing).ClearBox also relies on an oblivious server-aided key generation protocol to ensure that the stored files are encrypted with keys that are dependent on both the hash of the file and the gateway's secret.This protects against brute force search attacks when the message content is predictable, but also ensures that a curious gateway/storage provider which does not know the file hash cannot acquire the necessary keys to decrypt the file (since the key generation protocol is oblivious).To protect against malicious users who otherwise have obtained the file hash (e.g., by theft/malware) but do not possess the full file, ClearBox employs proofs of ownership over the encrypted file to verify that a given user is indeed in possession of the full file.

ClearBox 通过在访问内容时利用自行过期的 URL，对共享文件实施细粒度的访问控制。也就是说，每当用户希望访问给定的资源时，网关会动态地为该资源生成一个 URL，该 URL 会在一段短暂的时间后过期。如表 1 所示，现有的云 API 支持动态生成过期的 URL。通过这样做，ClearBox 不仅确保 G 可以限制对存储在云上的数据的访问，还使 G 能够跟踪其用户的访问模式(例如，用于计费)。ClearBox 还依赖于一个不经意的服务器辅助密钥生成协议，以确保存储的文件用依赖于文件散列和网关秘密的密钥加密。当消息内容是可预测的时，这防止了暴力搜索攻击，但也确保了不知道文件散列的好奇网关/存储提供者不能获得解密文件所需的密钥(因为密钥生成协议是不经意的)。为了防止恶意用户获得文件散列(例如，通过盗窃/恶意软件)但不拥有完整文件，ClearBox 使用加密文件的所有权证明来验证给定用户确实拥有完整文件。

In the following subsections, we go into greater detail on the various parts of ClearBox, starting with the building blocks that we will use in our solution, then moving on to the protocol specifica-tion, and finally to its security analysis.

在接下来的小节中，我们将更详细地介绍 ClearBox 的各个部分，首先是我们将在解决方案中使用的构建模块，然后是协议规范，最后是它的安全性分析。

3.2 Building Blocks

3.2 构建模块

Before describing ClearBox in detail, we start by outlining the building blocks that will be used in ClearBox.

在详细描述 ClearBox 之前，我们首先概述将在 ClearBox 中使用的构建块。

3.2.1 CARDIAC

3.2.1 心脏

Cryptographic accumulators (e.g. [12, 20-22, 30, 33, 39]) basi-cally constitute one-way membership functions;these functions can be used to answer a query whether a given candidate belongs to a set.

密码累加器(例如[12，20-22，30，33，39])基本上构成单向成员函数；这些函数可用于回答给定候选是否属于集合的查询。

In what follows, we show how to construct a cardinality-proving accumulator (CARDIAC) which leverages Merkle trees in order to efficiently provide proofs of membership and (non-public) proofs of maximum set cardinality.As we show in Section 3.3, proofs of cardinality are needed to attest the file deduplication patterns to users.

在接下来的内容中，我们将展示如何构建基数证明累加器(CARDIAC)，该累加器利用 Merkle 树来有效地提供成员资格证明和最大集合基数的(非公开)证明。正如我们在第 3.3 节中所展示的，需要基数证明来向用户证明文件重复数据消除模式。

A Merkle tree is a binary tree, in which the data is stored in the leaves.Let ai,j denote a node in the tree located at the i-th level and j-th position.Here, the level refers to the distance (in hops) to the leaf nodes;clearly, leaf nodes are located at distance 0.On the other hand, the position within a level is computed incrementally from left to right starting from position 0;for example, the leftmost node of level 1 is denoted by a1,0.In a Merkle tree, the intermediate nodes are computed as the hash of their respective child nodes;namely ai+1,j = H(ai,2j , ai,2j+1).

Merkle 树是一个二叉树，数据存储在树叶中。让 ai，j 表示树中位于第 I 层和第 j 个位置的节点。这里，级别指的是到叶节点的距离(以跳为单位)；显然，叶节点位于距离 0 处。另一方面，级别内的位置是从位置 0 开始从左到右递增计算的；例如，级别 1 最左边的节点由 a1，0 表示。在 Merkle 树中，中间节点被计算为它们各自子节点的散列；即 ai+1，j = H(ai，2j，ai，2j+1)。

Given a tree of height

给定一棵树的高度

, CARDIAC accumulates elements of a set X by assigning these to the leaf nodes (starting from position 0) while the remaining leaf nodes a0,|X|,...,a0,2

，通过将集合 X 的元素分配给叶节点(从位置 0 开始),而剩余的叶节点 a0，|X|，...，a0，2

-−1 are filled

-1 已填充

with a distinct symbol 0.We call these the zero leaves.Nodes that can be computed from the zero leaves play a special role.We denote these as open nodes in the sense that their values are openly known.More formally, the zero leaves a0,|X|,...,a0,2

带有独特的符号 0。我们称之为零叶。可以从零叶计算的节点起着特殊的作用。我们将它们称为开放节点，因为它们的价值是公开的。更正式地说，零留下 a0，|X|，...，a0，2

- are open.Moreover, if ai,2j and ai,2j+1 are both open, so is ai+1,j = H(i + 1, ai,2j , ai,2j+1).

-是开放的。而且，如果 ai，2j 和 ai，2j+1 都是开的，那么 ai+1 也是开的，j = H(i + 1，ai，2j，ai，2j+1)。

We now outline the main algorithms (Acc, ProveM, VerifyM, ProveC , VerifyC) provided by CARDIAC.Observe that ProveM and ProveC will be used to implement Attest and likewise VerifyM and VerifyM to instantiate Verify.

我们现在概述由 CARDIAC 提供的主要算法(Acc、ProveM、VerifyM、ProveC、VerifyC)。请注意，ProveM 和 ProveC 将用于实现证明，同样，VerifyM 和 VerifyM 也将用于实例化验证。

δ ← Acc(X).This algorithm accumulates the elements of a set X into a digest δ.In CARDIAC, δ corresponds to the hash of the root node of the modified Merkle tree, a

δ ← Acc(X)。该算法将集合 X 的元素累加成一个摘要 δ。在 CARDIAC 中，δ 对应于修改的 Merkle 树的根节点的散列，a

-,0, and the height

-，0，和高度

of the tree, i.e., δ = H(a

树的，即 δ = H(a

-,0,).

-,0,).

πM ← ProveM(X, x).Given a set X and element x ∈ X, this

πM ← ProveM(X，X)。给定集合 X 和元素 x ∈ X，这个

algorithm outputs a proof of membership πM asserting that x ∈ X. πM consists of the sibling path of x in the modified Merkle tree and the root a

算法输出隶属度 πM 的证明，断言 x ∈ X. πM 由修改的 Merkle 树中 x 的兄弟路径和根 a 组成

-,0.

-,0.

VerifyM(δ, x, πM).Given δ, an element x, its sibling path and

VerifyM(δ，x，πM)。给定 δ，元素 x，它的兄弟路径和

the root a

根 a

-,0, this algorithm outputs true if and only if δ = H(a

-，0，当且仅当 δ = H(a

-,0,) where

-，0)，其中

is the length of the sibling path and the sibling path of x matches the root a

是同级路径的长度，x 的同级路径与根 a 匹配

-,0.

-,0.

πC ← ProveC (X).Given a set X, this algorithm outputs a proof

πC ← ProveC (X)。给定集合 X，该算法输出一个证明

πC attesting an upper bound on the size |X| of the set.Here, πC consists of the size of X, the right-most non-zero element a0,|X|−1, its sibling path, and the root of the tree a

πC 证明集合的大小|X|的上限。这里，πC 由 X 的大小、最右边的非零元素 a0、| X |-1、它的兄弟路径和树 a 的根组成

-,0.

-,0.

VerifyC(δ, c, πC).Given the digest δ, the cardinality |X| of the

VerifyC(δ，C，πC)。在给定摘要 δ 的情况下

set X, and a proof of cardinality consisting of a0,|X|−1, its sibling path, and the root of the tree a

集合 X，以及由 a0、| X |-1、它的兄弟路径和树 a 的根组成的基数证明

-,0, this algorithm out-puts true if the following conditions are met:

-，0，如果满足以下条件，此算法输出为真:

• it holds that δ = H(a

它认为 δ = H(a

-,0,) where

-，0)，其中

is the length

长度是多少

of the sibling path,

兄弟路径的，

• it holds that 2−< |X| ≤ 2,

它认为 2 < | X |≤2，

• the sibling path of a0,|X|−1 matches the root a

a0，| X |-1 的兄弟路径与根 a 匹配

-,0, and • all nodes on the sibling path that are open do contain

-、0 和兄弟路径上所有打开的节点都包含

the right value.

正确的价值。

Here, VerifyC assumes that all the leafs with position larger than |X| − 1 in level 0 are filled with the 0 element which is not contained in X.

这里，VerifyC 假设在级别 0 中位置大于| X |-1 的所有叶子都用不包含在 X 中的 0 元素填充

In Appendix A, we show that, in addition to proofs of member-ship, our CARDIAC instantiation provides a proof that the number of non-zero leaves in the Merkle tree with root a

在附录 A 中，我们表明，除了成员资格的证明之外，我们的心实例化还提供了一个证明，证明了根为 A 的 Merkle 树中的非零叶的数量

-,0 is at most |X|.

-，0 最多是|X|。

3.2.2 Time-Dependent Randomness

3.2.2 与时间相关的随机性

ClearBox leverages a time-dependent randomness generator GetRandomness : T → {0, 1}seed where T denotes a set of dis-crete points in time.In a nutshell, GetRandomness produces val-ues that are unpredictable but publicly reconstructible.

ClearBox 利用了一个依赖于时间的随机性生成器 getrandomy:T →{ 0，1}seed，其中 T 表示一组离散的时间点。简而言之，获取随机性会产生不可预测但可公开重建的值。

More formally, let cur denote the current time.On input t ∈ T, GetRandomness outputs a uniformly random string in {0, 1}seed if t ≤ cur, otherwise GetRandomness outputs ⊥.We say that GetRandomness is secure if the output of GetRandomness(t) can-not be predicted with probability significantly better than 2−seed as long as t < cur.

更正式地说，让 cur 表示当前时间。在输入 t ∈ T 时，如果 t ≤ cur，则 get 随机性输出{0，1 }种子中的一致随机字符串，否则 get 随机性输出⊥.我们说，如果只要 t < cur，就不能用明显优于 2 种子的概率来预测 get 随机性(t)的输出，那么 get 随机性是安全的。

Similar to [10], we instantiate GetRandomness by leveraging functionality from Bitcoin, since the latter offers a convenient means (e.g., by means of API) to acquire time-dependent randomness.This condition minimizes the number of open nodes in the tree, and hence the effort in proving/verifying membership and cardi-nality.

类似于[10]，我们通过利用比特币的功能来实例化 GetRandomness，因为后者提供了一种方便的方法(例如，通过应用编程接口)来获取依赖于时间的随机性。这种情况使树中开放节点的数量最小化，从而减少了证明/验证成员资格和基数的工作。

891

891

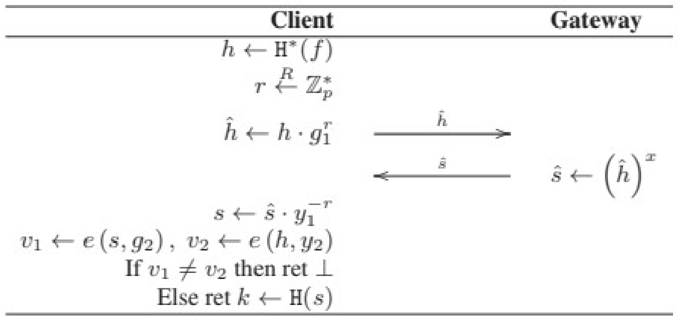

Client
Gateway
$h \leftarrow \mathtt{H}^*(f)$
$r \overset{R}{\leftarrow} \mathbb{Z}_p^*$
$\hat{h} \leftarrow h \cdot g_1^r$
$\hat{h}$
$\hat{s}$
$\hat{s} \leftarrow (\hat{h})^x$
$s \leftarrow \hat{s} \cdot y_1^{-r}$
$v_1 \leftarrow e(s, g_2),\ v_2 \leftarrow e(h, y_2)$
If $v_1 \neq v_2$ then ret $\perp$
Else ret $k \leftarrow \mathtt{H}(s)$

Figure 2: Server-aided key generation module based on blind BLS signatures.Here, Γ1, Γ2 are two groups with order p, g1, g2 generators of Γ1 and Γ2 respectively, the pairing func-tion e : Γ1 × Γ2 → ΓT , the hash function H: {0, 1}→ Γ1, the secret key x ∈ Zp, with corresponding public keys y1 = gx 1 , y2 = gx 2 .

图 2:基于盲 BLS 签名的服务器辅助密钥生成模块。这里，γ1、γ2 分别是 γ1 和 γ2 的 p 阶、g1 阶、g2 阶生成元的两组，配对函数 e:γ1×γ2→γT，散列函数 H: {0，1 }→γ1，秘密密钥 x ∈ Zp，对应的公钥 y1 = gx 1，y2 = gx 2。

Namely, Bitcoin relies on blocks, a hash-based Proof of Work con-cept, to ensure the security of transactions.In particular, Bitcoin peers need to find a nonce, which when hashed with the last block hash and the root of the Merkle tree accumulating recent transac-tions, the overall hash is smaller than a 256-bit threshold.

也就是说，比特币依靠区块(一种基于哈希的工作证明概念)来确保交易的安全性。具体来说，比特币对等体需要找到一个随机数，当用最后一个块哈希和累积最近交易的 Merkle 树的根进行哈希运算时，整体哈希小于 256 位阈值。

The difficulty of block generation in Bitcoin is adjusted so that blocks are generated once every 10 minutes on average;it was shown in [29] that the block generation in Bitcoin follows a shifted geometric distribution with parameter p = 0.19.Recent studies show that a public randomness beacon—outputting 64 bits of min-entropy every 10 minutes—can be built atop Bitcoin [8].

调整比特币中区块生成的难度，使区块平均每 10 分钟生成一次；在[29]中表明，比特币中的区块生成遵循参数 p = 0.19 的移位几何分布。最近的研究表明，可以在比特币上建立一个公共随机性信标——每 10 分钟输出 64 位最小熵[8]。

Given this, GetRandomness then unfolds as follows.On input time t, GetRandomness outputs the hash of the latest block that has appeared since time t in the Bitcoin block chain.Clearly, if t > cur corresponds to a time in the future, then GetRandomness will output ⊥, since the hash of a Bitcoin block that would appear in the future cannot be predicted.On the other hand, it is straight-forward to compute GetRandomness(t), for a value t ≤ cur (i.e., t is in the past) by fetching the hash of previous Bitcoin blocks.In this way, GetRandomness enables an untrusted party to sample randomness—without being able to predict the outcome ahead of time.

考虑到这一点，GetRandomness 然后展开如下。在输入时间 t 时，GetRandomness 输出自时间 t 以来在比特币区块链中出现的最新区块的散列。显然，如果 t > cur 对应于未来的某个时间，那么 get 随机性将输出⊥，因为未来出现的比特币块的散列是无法预测的。另一方面，对于 t ≤ cur 的值(即 t 是过去的值)，通过获取以前的比特币块的散列来直接计算 get 随机性(t)。通过这种方式，GetRandomness 使不可信的一方能够对随机性进行采样，而无法提前预测结果。

The purpose of GetRandomness is to ensure that the selection of files to which the gateway attests the deduplication pattern is randomly chosen and not precomputed.

get 随机性的目的是确保网关证明重复数据消除模式的文件选择是随机选择的，而不是预先计算的。

While using a time-dependent source of randomness is a viable option, other alternatives may be imaginable.For example, one may couple the selection of files with Fiat-Shamir heuristics [26] to ensure that the selection is randomly chosen from the point of view of the gateway and couple it with proofs of work.If the time effort to generate a valid proof of work is about the duration of an epoch, a malicious gateway may not be able to precompute the se-lections.We leave the question of investigating viable alternatives to external sources of randomness as an interesting direction for future research.

虽然使用依赖于时间的随机性来源是一个可行的选择，但其他选择也是可以想象的。例如，可以将文件的选择与菲亚特-沙米尔试探法[26]相结合，以确保从网关的角度随机选择选择，并将其与工作证明相结合。如果生成有效工作证明的时间大约是一个时期的持续时间，恶意网关可能无法预先计算选择。我们把研究外部随机性来源的可行替代方案作为未来研究的一个有趣方向。

3.2.3 Server-Aided Key Generation

3.2.3 服务器辅助密钥生成

ClearBox employs an oblivious protocol adapted from [14] which is executed between clients and the gateway to generate the keys required to encrypt the stored files.Unlike [14], our protocol does not rely on RSA, and is based on blind BLS signatures [17, 18].Although the verification of BLS signatures is more expensive than its RSA counterpart, BLS signatures are considerably shorter than RSA signatures, and are faster to compute by the gateway.

ClearBox 采用了一个改编自[14]的遗忘协议，该协议在客户端和网关之间执行，以生成加密存储文件所需的密钥。与[14]不同，我们的协议不依赖于 RSA，而是基于盲 BLS 签名[17，18]。虽然 BLS 签名的验证比 RSA 签名更昂贵，但 BLS 签名比 RSA 签名要短得多，而且网关计算速度更快。

As shown in Figure 2, we assume that at setup, the gateway chooses two groups $\Gamma 1$ and $\Gamma 2$ with order p, and a computable bi-

如图 2 所示，我们假设在设置时，网关选择两个组 γ1 和 γ2，顺序为 p，以及一个可计算的 bi-

linear map $e : \Gamma 1 \times \Gamma 2 \rightarrow \Gamma T$. Additionally, the gateway chooses a private key $x \in Zp$, and the corresponding public keys $y1 = gx\ 1 \in \Gamma 1$ and $y2 = gx\ 2 \in \Gamma 2$.Let $H: \{0, 1\} \rightarrow \Gamma 1$ be a crypto-graphic hash function which maps bitstrings of arbitrary length to group elements in $\Gamma 1$.Prior to storing a file f, the client computes $h \leftarrow H(f)$, blinds it by multiplying it with gr 1, given a randomly chosen $r \in Zp$, and sends the blinded hash hˆ to the gateway.The latter derives the signature on the received message and sends the result back to the client, who computes the unblinded signature s and verifies that: e (s, g2) = e hxgrx 1 g−rx 1 , g2 = e (h, y2).

线性映射 e:γ1×γ2→γT 另外，网关选择一个私钥 x ∈ Zp，对应的公钥 y1 = GX 1∈γ1 和 y2 = GX 2∈γ2。设 H: {0，1 }→γ1 是一个加密哈希函数，它将任意长度的位串映射到 γ1 中的组元素。在存储文件 f 之前，客户端计算 h ← H(f)，给定随机选择的 r ∈ Zp，通过将其乘以 gr 1 来隐藏它，并将隐藏的哈希 H 发送到网关。后者根据接收到的消息导出签名，并将结果发送回客户端，客户端计算未隐藏的签名 s，并验证:e (s，G2)= e hxgrx 1g rx 1，g2 = e (h，y2)。

The encryption key is then computed as the hash of the unblinded signature: $k \leftarrow H(s)$.The benefits of such a key generation module are twofold:

然后，加密密钥被计算为未加密签名的散列:k ← H(s)。这种密钥生成模块有两个好处:

• Since the protocol is oblivious, it ensures that the gateway does not learn any information about the files (e.g., about the file hash) during the process.On the other hand, this proto-col enables the client to check the correctness of the compu-tation performed by the gateway (i.e., verify the gateway's signature).As we show later, this verification is needed to prevent a rational G from registering users of the same file to different file versions with reduced level of deduplication.

由于协议是不经意的，它确保网关在过程中不了解任何关于文件的信息(例如，关于文件散列)。另一方面，这个原型使客户端能够检查网关执行的计算的正确性(即验证网关的签名)。正如我们稍后所展示的，需要进行这种验证，以防止 rational G 将同一文件的用户注册到重复数据消除级别降低的不同文件版本。

• By involving the gateway in the key generation module, brute-force attacks on predictable messages can be slowed down by rate-limiting key-generation requests to G. Notice that, similar to [14], this scheme does not prevent a curious G from performing brute-force searches on predictable mes-sages, acquiring the hash, and the corresponding key k. In this sense, the security offered by our scheme reduces to that of existing MLE schemes (cf. Section 3.4).

通过将网关包含在密钥生成模块中，对可预测消息的暴力攻击可以通过对 G 的速率限制密钥生成请求来减缓。请注意，与[14]类似，该方案不会阻止好奇的 G 对可预测消息执行暴力搜索，获取哈希和相应的密钥 k。从这个意义上说，我们的方案提供的安全性降低到现有 MLE 方案的安全性(参见第 3.4 节)。

3.2.4 Proof of Ownership

3.2.4 所有权证明

The aforementioned server-aided key generation ensures that an adversary which is not equipped with the correct file hash cannot acquire the file encryption key k. However, a user who has wrong-fully obtained the file hash would be able to claim file ownership.In this case, Proofs of Ownership (PoW) can be used by the gate-way to ensure that the client is in possession of the file in its entirety (and not only of its hash) [16, 23, 27].

前述服务器辅助密钥生成确保了没有配备正确文件散列的对手不能获得文件加密密钥 k。然而，错误地完全获得文件散列的用户将能够主张文件所有权。在这种情况下，gate-way 可以使用所有权证明(PoW)来确保客户端拥有整个文件(而不仅仅是其散列)[16，23，27]。

In this paper, we rely on the PoW due to Halevi et al. [27] which, as far as we are aware, results in the lowest computational and stor-age overhead on the gateway [16].This PoW computes a Merkle tree over the file f, such that the root of the Merkle tree constitutes a commitment to the file.In the sequel, we denote by MTBuf(f) the root of the Merkle tree as output by the PoW of [27] given an input Buf(f), being an encoding of a file f. The verifier can challenge the prover for any block of the file, and the prover is able to prove knowledge of the challenged block by submitting the authentica-tion path of this block.In order to reduce the number of challenges for verifying the PoW, the file is encoded into Buf(f) before com-puting the Merkle tree.An additional trade-off can be applied by limiting Buf(f) to a maximum size of 64 MB in the case of larger files.

在本文中，我们依赖于 Halevi 等人[27]的 PoW，据我们所知，这导致网关的计算和存储开销最低[16]。这个 PoW 在文件 f 上计算一个 Merkle 树，这样 Merkle 树的根就构成了对文件的承诺。在后续文章中，我们用 MTBuf(f)表示 Merkle 树的根，作为[27]的 PoW 给定输入 Buf(f)的输出，是文件 f 的编码。验证者可以针对文件的任何块向证明者提出质疑，证明者能够通过提交该块的认证路径来证明知道被质疑的块。为了减少验证 PoW 的挑战，在计算 Merkle 树之前，文件被编码成 Buf(f)。在文件较大的情况下，可以通过将 Buf(f)限制为 64 MB 的最大大小来进行额外的权衡。

The security of this scheme is based on the minimum distance of random linear codes.We refer the readers to Appendix B for more details on the PoW of [27].

该方案的安全性基于随机线性码的最小距离。关于[27]的功率的更多细节，我们请读者参考附录 B。

3.3 ClearBox: Protocol Specification

3.3 ClearBox:协议规范

We now detail the specifications for the procedures of ClearBox.As mentioned earlier, we assume in the sequel that G owns an ac-count hosted by S, and that the communication between C, S, and G occurs over authenticated and encrypted channels.

我们现在详细说明 ClearBox 程序的规范。如前所述，我们在续集中假设 G 拥有一个由 S 托管的 ac-count，并且 C、S 和 G 之间的通信通过经过身份验证和加密的通道进行。

892

892

Specification of the Put Procedure.

卖出程序的说明。

In ClearBox, when a client C wishes to upload a new file f onto S, C issues an upload request to G. Subsequently, C and G start executing the server-aided key generation protocol described in Section 3.2.3.The outcome of that protocol is the key k ← H(s), where s ← H(f) x given a cryptographic hash function H.

在 ClearBox 中，当客户端 C 希望将新文件 f 上传到 S 上时，C 向 G 发出上传请求。随后，C 和 G 开始执行第 3.2.3 节中描述的服务器辅助密钥生成协议。该协议的结果是密钥 k ← H(s)，其中 s ← H(f) x 给定一个加密散列函数 H。

C then encrypts f using encryption algorithm enc under key k, computes and sends to G the root of the Merkle tree output by the PoW of [27], that is FID ← MTBuf(enc(k,f)) where Buf is an encoding function (cf. Section 3.2.4).

c 随后使用密钥 k 下的加密算法 enc 对 f 进行加密，计算并向 G 发送[27]的 PoW 输出的 Merkle 树的根，即 FID ← MTBuf(enc(k，f))，其中 Buf 是一个编码函数(参见第 3.2.4 节)。

Subsequently, G checks if any other client has previously stored a file indexed by FID.Here, two cases emerge:

随后，G 检查是否有任何其他客户端以前存储过由 FID 索引的文件。这里出现了两种情况:

f has not been stored before: In this case, G issues a timed gen-

f 以前没有存储过:在这种情况下，G 发出一个定时的 gen-

erateURL command allowing the client to upload the data onto G's account within a time interval.Recall that a timed generateURL command results in a URL which expires af-ter the specified period of time.After the upload of the encrypted file f ← enc(k, f) terminates, G accesses S, computes MTBuf(f) using the PoW of [27], and verifies that it matches FID.If the verification matches, G stores the metadata associated with f in a newly generated struc-ture indexed by FID (such as the client ID C and the size of f , see Section 4 for more details).Otherwise, if MTBuf(f) does not match FID, G deletes the file and appends C to a blacklist.

允许客户端在一定时间间隔内将数据上传到 G 的账户上的命令。回想一下，定时生成器命令会产生一个在指定时间段后过期的网址。在加密文件 f ← enc(k，f)的上传终止后，G 访问 S，使用[27]的 PoW 计算 MTBuf(f)，并验证其与 FID 匹配。如果验证匹配，G 将与 f 相关联的元数据存储在由 FID 索引的新生成的结构中(例如客户端 ID C 和 f 的大小，更多细节请参见第 4 节)。否则，如果 MTBuf(f)与 FID 不匹配，G 将删除该文件并将 C 追加到黑名单中。

f has been stored before: In this case, G requests that C proves

f 以前存储过:在这种情况下，G 请求 C 证明

that it owns the file f. For that purpose, G and C execute the PoW protocol of [27] (we refer the reader to Appendix B for more details).In essence, G chooses a random number u of leaf indexes of the Merkle tree computed over Buf(f), and asks C for for the sibling-paths of all the u leaves.In response, C returns the sibling paths corresponding to the chosen u leafs associated with the Merkle tree of Buf(f).G accepts if all the sibling paths are valid with respect to the stored FID.If this verification passes, G appends C to the file structure FID, and sends an ACK to C. In turn, C deletes the local copy of the file, and only needs to store FID and the key k.

为此，G 和 C 执行[27]的 PoW 协议(更多细节请读者参考附录 B)。本质上，G 选择一个在 Buf(f)上计算的 Merkle 树的叶索引的随机数 u，并要求 C 给出所有 u 个叶的兄弟路径。作为响应，C 返回对应于与 Buf(f)的 Merkle 树相关联的所选 u 叶的兄弟路径。如果相对于存储的 FID，所有的兄弟路径都有效，g 接受。如果这个验证通过，G 将 C 追加到文件结构 FID 中，并向 C 发送 ACK，反过来，C 删除文件的本地副本，只需要存储 FID 和密钥 k。

Specification of the Get Procedure.

获取程序规范。

To download a file with index FID, C submits FID to G;the latter checks that C is a member of the user list added to the meta-data structure of FID.If so, G generates a timed URL allowing C to download the requested file from S.

要下载带有索引 FID 的文件，C 向 G 提交 FID；后者检查 C 是否是添加到 FID 的元数据结构中的用户列表的成员。如果是这样，G 生成一个定时 URL，允许 C 从 s 下载请求的文件。

Notice that if C did not locally cache the decryption key associ-ated with FID, then C can leverage its knowledge of H(f) in or-der to acquire the corresponding key by executing the server-aided generation protocol with G.

请注意，如果 C 没有在本地缓存与 FID 相关联的解密密钥，那么 C 可以利用其对 H(f)的了解，通过用 g 执行服务器辅助生成协议来获取相应的密钥

Specification of the Delete Procedure.

删除程序的说明。

When C wants to delete file FID, it informs G. G marks C for removal from the metadata structure associated with FID in the subsequent epoch (see Section 3.3 for more details).If no further clients are registered for this file, G sends a request to S to delete it.

当 C 想要删除文件 FID 时，它会通知 G. G 标记 C，以便在随后的时期中从与 FID 相关联的元数据结构中删除(更多详细信息，请参见第 3.3 节)。如果没有其他客户端注册该文件，G 会向 S 发送删除请求。

In our implementation, we set the expiry timeout of the URL to 30 seconds;this means that clients have 30 seconds to start the upload process.

在我们的实现中，我们将 URL 的到期超时设置为 30 秒；这意味着客户端有 30 秒的时间开始上传过程。

Here, G additionally notes the number of download requests per-formed by C for file FID.

在这里，G 还记录了由 C 为文件 FID 形成的下载请求的数量。

Specification of the Attest Procedure.

证明程序的说明。

At the end of each epoch, G attests the deduplication patterns of its clients' files, e.g., in their bills.Notice that if a client re-quested the deletion of a file f during the epoch, G only removes the marked clients from the clients list subscribed to FID after the end of the epoch.

在每个时期结束时，G 会证明其客户文件的重复数据消除模式，例如在他们的账单中。请注意，如果一个客户端在纪元期间请求删除文件 f，则在纪元结束后，G 只会从订阅 FID 的客户端列表中删除标记的客户端。

At the end of epoch Ej , the bill of every client C of G includes for each stored file f the number of accesses by C and the cardinal-ity of CFID , which denotes the set of clients registered to f. Here, we assume a static setting where clients are charged for files that they are storing within each epoch;this conforms with the func-tionality of existing providers, such as Amazon S3, which rely on fixed epochs for measuring storage consumption (e.g., the epoch interval is 12 hours in Amazon S3).Since our Attest procedure is efficient (cf. Section 4), our solution can be made practically dynamic by relying on extremely small epochs—in which case the bill can accumulate the fine-grained storage consumption over a number of epochs.

在纪元 Ej 结束时，G 的每个客户端 C 的账单包括每个存储文件 f 的 C 的访问次数和 CFID 基数，该基数表示注册到 f 的客户端集合。这里，我们假设一个静态设置，其中客户端为它们在每个纪元内存储的文件收费；这符合现有提供商的功能，如亚马逊 S3，它们依赖固定的纪元来衡量存储消耗(例如，在亚马逊 S3，纪元间隔为 12 小时)。由于我们的证明过程是高效的(参见第 4 节)，我们的解决方案可以通过依赖极小的时期而变得实际动态，在这种情况下，账单可以在多个时期内累积细粒度的存储消耗。

The bill issued by the provider acts as a binding commitment by G to the deduplication (and access) patterns of its clients' files.Af-ter the bills are issued, G needs to convince his clients that they are correctly formed.Notice that G keeps a record of the authenticated download requests by clients, which can be used to prove the num-ber of their accesses per file.Moreover, G needs to prove to all clients of a file f that:

提供商出具的账单是 G 对其客户文件的重复数据消除(和访问)模式的约束性承诺。票据发行后，G 需要让他的客户相信票据是正确的。请注意，G 保存了客户端验证下载请求的记录，可以用来证明每个文件的访问次数。此外，G 需要向文件 f 的所有客户证明:

• Each of these clients is included in the set of clients CFID storing FID.

这些客户中的每一个都包含在 CFID 存储 FID 的客户集中。

• The size of CFID is as claimed in the bill.• Clients storing f are referenced to the same set CFID .As described in Section 3.4, this last condition prevents G from including users of same file into different accumulators, resulting in the under-reporting of the file deduplication patterns.

CFID 的面积与法案中宣称的一样大。存储 f 的客户端引用同一个 CFID 集。如第 3.4 节所述，最后一种情况会阻止 G 将同一文件的用户包含到不同的累加器中，从而导致文件重复数据消除模式的报告不足。

The first two proofs can be realized by G using CARDIAC.More specifically, G accumulates the IDs of the clients storing FID;here, for each user Ci ∈ CFID , a leaf node with value Ui ← H(FID|| Ci||Ej ||seedi,j) is created, where seedi,j denotes a nonce which is sampled for this client and this epoch and which is com-municated within the bill.As we show in Section 3.4, this protects user privacy and ensures that the ID of any user is not given in the clear to other users when G issues proofs of size and/or mem-bership.The accumulation of all Ui ∈ CFID results into a digest denoted by δ.As described earlier, CARDIAC enables G to prove to each client in the set that (i) the client is part of the accumu-lated set CFID and (ii) an upper bound of the size |CFID | of the accumulated set.Observe that an upper bound on the cardinality is sufficient to protect against a rational gateway.This is the case since the gateway does not benefit from reporting a larger |CFID |, since this might entail price reductions to the clients storing f.

前两个证明可以通过 G 使用 CARDIAC 来实现。更具体地说，G 累积存储 FID 的客户端的标识；这里，对于每个用户 Ci ∈ CFID，创建值为 Ui ← H(FID||Ci||Ej ||seedi，j)的叶节点，其中 seedi，j 表示为此客户端和此纪元采样的随机数，并且该随机数在账单内通信。正如我们在第 3.4 节中所展示的，这保护了用户隐私，并确保了当 G 发布大小和/或内存证明时，任何用户的 ID 都不会清楚地提供给其他用户。所有 Ui ∈ CFID 的累积产生一个用 δ 表示的摘要。如前所述，心动图使 G 能够向集合中的每个客户证明(I)客户是累积集合 CFID 的一部分，以及(ii)累积集合的大小|CFID |的上限。请注意，基数的上限足以防范 rational gateway。这是因为网关不能从报告更大的|CFID |中获益，因为这可能会降低存储 f 的客户端的价格。

Figure 3 shows an example for a set comprising 5 clients U0,..., U4.The elements circled in dotted red depict the membership proof for U2, while the elements circled in solid blue depict the proof of set cardinality which consists of the sibling path of the last non-zero element.The grey elements consist of the open nodes, i.e., the zero leaves which are not associated to any client and hence filled with symbol 0, and all nodes that can be derived from these.Note that the proofs of membership and set cardinality can be further compressed in case the respective sibling paths contain the same elements.

图 3 示出了包括 5 个客户端 U0 的集合的例子，...，U4。用红色虚线圈出的元素描述了 U2 的成员证明，而用蓝色实线圈出的元素描述了集合基数的证明，该集合基数由最后一个非零元素的兄弟路径组成。灰色元素包括开放节点，即不与任何客户端相关联的零叶，因此用符号 0 填充，以及可以从这些节点导出的所有节点。请注意，如果相应的兄弟路径包含相同的元素，则可以进一步压缩成员资格和集合基数的证明。

G still needs to show that all the clients storing FID are accumu-lated in the same accumulator.This can be achieved by publishing the association between FID, and its corresponding accumulator digest on a public bulletin board (e.g., on the public website of G).However, this approach does not scale with the number of files

g 仍然需要显示所有存储 FID 的客户端都累积在同一个累加器中。这可以通过在公共公告板上(例如，在 G 的公共网站上)发布 FID 与其对应的累加器摘要之间的关联来实现。但是，这种方法不会随着文件数量的增加而扩展

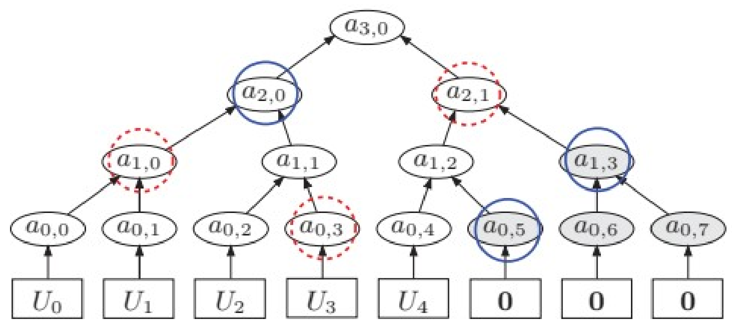


Figure 3: Sketch of a proof of set membership and set cardinal-ity given a set of 5 elements in CARDIAC.

图 3:在心动图中给定一组 5 个元素的情况下，集合成员和集合基数的证明草图。

processed by G. For that reason, we elect to only publish those associations for a randomly chosen subset of all files.Here, G first obtains a random seed by invoking GetRandomness(tj), where tj denotes a point in time after the issuance of the bills for epoch Ej which is determined by a deterministic and publicly known process.This seed is used to sample the files whose accumulators need to be published during this epoch;for instance, G can extract ν-bits of the seed and use them as a mask to decide which file accumulators to publish.The probability that any file FID is selected at the end of a given epoch is subsequently given by 2−ν.

出于这个原因，我们选择只发布随机选择的所有文件子集的关联。这里，G 首先通过调用 GetLayments(tj)获得一个随机种子，其中 TJ 表示纪元 Ej 的票据发行之后的时间点，该时间点由确定性的和众所周知的过程确定。此种子用于对累加器需要在此时期发布的文件进行采样；例如，G 可以提取种子的 ν 位，并将其用作掩码来决定发布哪个文件累加器。在给定时期结束时选择任何文件 FID 的概率随后由 2ν 给出。

Notice that, in this case, G needs to compute proofs of member-ship and set cardinality only for the selected files;this information is sent to each client storing the selected files.The pairs (FID, δ) for the selected files are subsequently published by G (e.g., on its own website).

请注意，在这种情况下，G 需要计算成员资格的证明，并且只为选定的文件设置基数；该信息被发送到存储所选文件的每个客户端。所选文件的配对(FID，δ)随后由 G 发布(例如，在其自己的网站上)。

Specification of the Verify Procedure.

验证程序的规范。

The Verify procedure is only conducted by clients storing files for which the corresponding pairs (FID, δ) have been published by G at the end of the epoch.Notice that clients can easily check which FID are sampled by invoking GetRandomness(tj).

验证过程仅由存储文件的客户端执行，文件对应对(FID，δ)已由 G 在纪元结束时发布。请注意，客户端可以通过调用 GetLaminess(TJ)轻松检查哪些 FID 被采样。

Given the proofs of membership and set cardinality issued by G for their files, clients invoke the VerifyC and VerifyM algorithms of CARDIAC in order to verify that their ID is included in the set of clients CFID storing FID, and that |CFID | is as claimed in the bill.Moreover, clients check the pairs (FID, δ) published by G to verify that there is only one set CFID which they are referenced to.

给定由 G 为他们的文件发布的成员资格和集合基数的证明，客户端调用 CARDIAC 的 VerifyC 和 VerifyM 算法，以验证他们的 ID 包含在 CFID 存储 FID 的客户端集合中，并且|CFID |如账单中所述。此外，客户检查由 G 发布的对(FID，δ)，以验证他们参考的只有一个 CFID 集。

In this case, the proof of membership consists of the sibling-paths of leaves of the tree of height log2(|CFID |) .Hence, the size of the membership proof is at most 2 log2(|CFID |) hash values and verification takes at most 2 log2(|CFID |) executions of the hash function.On the other hand, the proof of cardinal-ity requires at most 2−hash operations to check the open leaves (since we assume that at least half of the leaves are occupied by the clients);that is, the size of the proof of cardinality is at most 2 log2(|CFID |) hash values, but verification requires O(|CFID |) hash operations.Notice that this is at most half the effort required to construct δ in CARDIAC.As shown in Section 4, this process is extremely efficient;notice that the verification of VerifyC and VerifyM can be made even more efficient if the client pre-computes and stores the open (zero) leaves in the tree.As the values of these nodes are independent of the file ID and the IDs of the clients, this precomputation could be performed once for a selection of trees of different heights.

在这种情况下，成员关系的证明由高度为 log2(|CFID |)的树的叶子的兄弟路径组成。因此，成员证明的大小最多为 2 个 log2(|CFID |)哈希值，验证最多需要 2 个 log2(|CFID |)哈希函数的执行。另一方面，基数证明最多需要 2 次散列操作来检查未结的叶子(因为我们假设至少一半的叶子被客户端占用)；也就是说，基数证明的大小最多是 2 个 log2(|CFID |)哈希值，但是验证需要 O(|CFID |)哈希操作。请注意，这最多是在 CARDIAC 中构建 δ 所需工作的一半。如第 4 节所示，该过程效率极高；请注意，如果客户端预先计算并存储树中的开放(零)叶子，VerifyC 和 VerifyM 的验证可以变得更加有效。由于这些节点的值独立于文件标识和客户端标识，因此可以对不同高度的树进行一次预计算。

Additional Operations.

附加操作。

Directories and other functionality: ClearBox hides the clients' directory structures from G by working on a single directory struc-ture hosted within S's account on the cloud.This has the benefit of

目录和其他功能:ClearBox 通过处理托管在云上的用户帐户中的单个目录结构，对用户隐藏了客户的目录结构。这样做的好处是

reducing the overhead borne by G (i.e., no path related overhead) and minimizes information leakage towards G.

减少了 G 承担的开销(即没有与路径相关的开销)，并最大限度地减少了向 G 的信息泄漏

Directory operations such as directory creation, directory renam-ing, etc.are locally handled by the software client of the users.Here, local directories contain pointers to the files stored therein and outsourced to the cloud;this enables the local client to perform operations such as directory listing and file renaming without the need to contact G—thereby minimizing the overhead incurred on G. Only operations that affect the client files stored on the cloud (e.g., file deletion/creation) are transmitted to G. Other APIs: Recall that ClearBox leverages expiring URL-based PUT commands (exposed by Amazon S3 and Google Cloud Stor-age [3]) to enable clients to upload new objects directly to S;ex-piring URLs are also important in ClearBox to revoke data access to clients.

目录操作，如目录创建、目录重命名等。由用户的软件客户端本地处理。这里，本地目录包含指向存储在其中并外包给云的文件的指针；这使得本地客户端无需联系 G 即可执行目录列表和文件重命名等操作——从而最大限度地减少了 G 产生的开销。只有影响存储在云上的客户端文件的操作(例如，文件删除/创建)才会传输给 G。其他 API:回想一下，ClearBox 利用即将到期的基于 URL 的 PUT 命令(由 Amazon S3 和 Google Cloud Stor-age [3]公开)使客户端能够将新对象直接上传到 S；在 ClearBox 中，取消对客户端的数据访问也很重要。

A number of commodity cloud service providers such as Drop-box, and Google drive, however, do not support URL commands for file creation, and only provide (non-expiring) URL-based file download.To integrate ClearBox with such commodity storage providers, we note the following differences to the protocol speci-fication of ClearBox (cf. Section 3.3):

然而，许多商品云服务提供商，如 Drop-box 和 Google drive，不支持用于文件创建的 URL 命令，并且只提供基于(未过期的)URL 的文件下载。为了将 ClearBox 与此类商品存储提供商集成，我们注意到 ClearBox 协议规范的以下差异(参见第 3.3 节):

• At file upload time, the URL-based PUT is replaced by the clients uploading the file to G, which in turn uploads the file to S. Recall that G has to compute the Merkle tree over the uploaded file;this can be done asynchronously before G uploads the file to S—therefore reducing the performance penalty incurred on G.

在文件上传时，基于网址的 PUT 被将文件上传到 G 的客户端所取代，G 再将文件上传到 s。回想一下，G 必须计算上传文件上的 Merkle 树；这可以在 G 将文件上传到 S 之前异步完成，因此减少了 G 的性能损失。

• Files are stored under random identifiers, and can be ac-cessed by means of permanent URLs which map to the file ID.When the user requests to delete a file, G renames the file to a new randomly selected ID.Other legitimate clients who require access to the file have to contact G who informs them of the new URL corresponding to the renamed file object.

文件存储在随机标识符下，可以通过映射到文件标识的永久网址进行访问。当用户请求删除文件时，G 会将文件重命名为新的随机选择的 ID。其他需要访问文件的合法客户端必须联系 G，G 会通知他们与重命名的文件对象相对应的新 URL。

In Section 4, we present a prototype implementation of ClearBox which interfaces with Dropbox, and we use it to evaluate the per-formance of this alternative technique.

在第 4 节中，我们展示了一个与 Dropbox 接口的 ClearBox 的原型实现，并使用它来评估这种替代技术的性能。

Rate-Limiting: Similar to [14], we rate-limit requests to G to pre-vent possible brute-force search attacks on predictable file contents (in the assisted key generation phase), and to prevent resource ex-haustion attacks by malicious clients.For this purpose, we limit the number of requests Ri that a client can perform in a given time frame Ti to a threshold θmax.

速率限制:类似于[14]，我们对 G 的请求进行速率限制，以防止对可预测文件内容的暴力搜索攻击(在辅助密钥生成阶段)，并防止恶意客户端的资源耗尽攻击。为此，我们将客户端在给定时间帧 T1 内可以执行的请求数 Ri 限制在阈值 θmax 内。

3.4 Security Analysis

3.4 安全性分析

In this section, we address the security of ClearBox with respect to the model outlined in Section 2.

在本节中，我们将针对第 2 节中概述的模型讨论 ClearBox 的安全性。

Secure Access: We start by showing that ClearBox ensures that a client who uploaded a file with Put will be able to recover the file with Get as long as he does not delete it.Since we assume that G and S will not tamper with the storage protocol, the threat to the soundness argument can only originate from other malicious clients.Moreover, since the procedures Get and Verify do not mod-ify the stored data, we will focus the analysis on the operations Delete and Put.

安全访问:我们从展示 ClearBox 确保用 Put 上传文件的客户只要不删除文件，就能用 Get 恢复文件开始。因为我们假设 G 和 S 不会篡改存储协议，所以对健全性论点的威胁只能来自其他恶意客户端。此外，由于获取和验证过程不会修改存储的数据，因此我们将重点分析删除和放入操作。

Clearly, the Delete procedure only removes the access of the user requesting the deletion.Impersonation is not possible here since we assume a proper identity management by G. Namely, the gateway will only delete a file with ID FID if |CFID | = 0.

显然，删除过程只是删除了请求删除的用户的访问权限。在这里模拟是不可能的，因为我们假设 g 进行了正确的身份管理。也就是说，只有当|CFID | = 0 时，网关才会删除带有标识 FID 的文件。

On the other hand, the Put procedure can only incur in the mod-ification of data when a file f is uploaded for the first time.A subsequent upload of f is deduplicated and therefore does not en-

另一方面，Put 过程只能在第一次上传文件时导致数据修改。f 的后续上载经过重复数据消除，因此不会-

894

tail any data modification at S. Notice that during initial upload, a malicious client can try to defect from the protocol, and construct an FID that does not fit to f, upload another file, or encrypt the file using the wrong key, etc.Recall that G verifies FID by down-loading the uploaded file f to check whether FID ?= MTBuf(f*).Notice that if a malicious user creates a malformed f , that is f = enc(H(H(f) x), f) for any file f, then this will result into a random FID value which, with overwhelming probability, will not collide with any other file ID.That is, f would be stored by S without being deduplicated—which incurs storage costs on the malicious client (as he is fully charged for storing f) without af-fecting the remaining honest clients.

在 s 处跟踪任何数据修改。请注意，在初始上传期间，恶意客户端可能会试图脱离协议，并构建不适合 f 的 FID，上传另一个文件，或者使用错误的密钥加密文件等。回想一下，G 通过下载上传的文件 f 来验证 FID，以检查 FID 是否？ = MTBuf(f*)。请注意，如果恶意用户为任何文件 f 创建了一个格式错误的

f，即 f = enc(H(f(f)x)，f)，那么这将导致一个随机的 FID 值，该值很可能不会与任何其他文件 ID 冲突。也就是说，f 将由 S 存储，而不进行重复数据消除—这将导致恶意客户端的存储成本(因为他存储 f 的费用是全额收费的)，而不会影响其余诚实的客户端。

With respect to illegitimate client access, we distinguish between (i) the case that a client obtains ownership of a file when uploading the file for the first time with Put and (ii) the case where the client accesses the file with Get at some later point in time without having the ownership in the file.For case (i), uploading of a file that is not yet stored will not help the client.Namely, the client cannot pretend to upload a file with a different FID as G will check the integrity of the file.When a user requests to upload a file which has been stored already, the adopted proof of ownership scheme (PoW) (cf. Section 3.2.4 and Appendix B) ensures that the client must know the entire file, or the corresponding encoded data buffer (in the case of larger files) which is larger than an assumed leakage threshold.More details of the security of the PoW scheme can be found in Appendix B. In case (ii), when a client requests access to f, G first checks if this client is still subscribed to f. Observe that this verification is handled internally by the gateway—without giving clients the ability to interfere with this process.If access is granted, the client gets a temporary URL which expires after a short time.Thus, this information cannot be re-used for further accesses;in particular, clients that are no longer subscribed to this file are not able to access the file any further.

关于非法客户端访问，我们区分(I)客户端在首次使用 Put 上传文件时获得文件所有权的情况和(ii)客户端在稍后某个时间点使用 Get 访问文件而不拥有文件所有权的情况。对于情况(I)，上传尚未存储的文件对客户端没有帮助。也就是说，客户端不能假装上传带有不同 FID 的文件，因为 G 将检查文件的完整性。当用户请求上传已经存储的文件时，采用的所有权证明方案(PoW)(参见第 3.2.4 节和附录 B)确保客户端必须知道整个文件，或者知道大于假设泄漏阈值的相应编码数据缓冲区(在文件较大的情况下)。关于 PoW 方案安全性的更多细节可以在附录 b 中找到。在情况(ii)中，当客户端请求访问 f 时，G 首先检查该客户端是否仍然订阅 f。请注意，该验证是由网关内部处理的，而不会给客户端干扰该过程的能力。如果授予访问权限，客户端将获得一个临时网址，该网址将在短时间后过期。因此，该信息不能被再次用于进一步的访问；特别是，不再订阅该文件的客户端无法再访问该文件。

Notice that also external adversaries cannot acquire access to f due to reliance on authenticated channels.Namely, we assume that all exchanged messages are appropriated signed, thus no repudiation of requests will be possible.

请注意，由于依赖经过身份验证的通道，外部对手也无法访问 f。也就是说，我们假设所有交换的消息都经过适当的签名，因此不可能拒绝请求。

Secure Attestation: Let CFID denote the set of clients that are storing file f.

安全证明:让 CFID 表示存储文件 f 的一组客户端

Recall that f is referenced with FID ← MTBuf(enc(k,f)) where k ← H (H(f) x) and Buf denotes the encoding used in PoW.The key k is deterministically computed from the hash H(f) of the file according to our server-assisted encryption scheme.Due to the collision resistant hash function used throughout the process, FID is uniquely bound to f. Therefore, all clients that are registered to f are likewise expecting the same ID FID.

回想一下，f 是用 FID ← MTBuf(enc(k，f))引用的，其中 k ← H (H(f) x)和 Buf 表示 PoW 中使用的编码。根据我们的服务器辅助加密方案，从文件的散列 H(f)中确定性地计算密钥 k。由于在整个过程中使用了防冲突的散列函数，FID 唯一地绑定到 f。因此，注册到 f 的所有客户端都期望相同的 ID FID。

According to the work flow of ClearBox, each client Ci ∈ CFID is informed about the size of CFID referencing to the according file ID FID.Therefore, G first commits to the pair FID, |CFID |.Sub-sequently, G samples a subset of files for which it proves the cor-rectness of the deduplication patterns using the accumulator.For these files the gateway G publishes the association between FID and the digest δ of the corresponding accumulator.This sampling is enforced by the GetRandomness function which acts as an un-predictably time-dependent source of randomness.This gives neg-ligible advantage for G in learning the files that will be sampled when committing to |CFID | at the end of the epoch.As the out-put of GetRandomness can be verified by the clients, each client Ci ∈ CFID can check if G has reported δ for the corresponding FID.

根据 ClearBox 的工作流程，参考相应的文件 ID FID，每个客户 Ci ∈ CFID 都被告知 CFID 的大小。因此，G 首先承诺对 FID，|CFID |。随后，G 对文件子集进行采样，并使用累加器证明重复数据消除模式的正确性。对于这些文件，网关 G 公布 FID 和相应累加器的摘要 δ 之间的关联。这种采样是由 GetRandomness 函数强制执行的，该函数充当不可预测的依赖于时间的随机性源。这使得 G 在学习当在纪元结束时提交给|CFID |时将被采样的文件方面具有不可忽视的优势。由于客户机可以验证 get 随机性的输出，每个客户机 Ci ∈ CFID 可以检查 G 是否报告了相应 FID 的 δ。

Following from the security of CARDIAC (cf. Appendix A), G can only prove set membership and cardinality to its clients, if

根据 heart 的安全性(参见附录 A)，G 只能向其客户证明集合成员和基数，如果

the underlying set and δ were computed correctly.Recall that δ constitutes a commitment to the set CFID .By publishing a list of associations (FID, δ) for the sampled file IDs, clients can ensure that G did not split CFID into separate subgroups.We refer the readers to Appendix A for a security treatment of CARDIAC.

基础集和 δ 计算正确。回想一下，δ 是对设定的 CFID 的承诺。通过发布采样文件标识的关联列表(FID，δ)，客户端可以确保 G 没有将 CFID 分成单独的子组。关于心脏疾病的安全治疗，我们请读者参考附录 A。

Data Confidentiality: As explained in Section 2, the knowledge of files size and user access patterns is essential for G and S to operate their business.Hence, hiding this information cannot be achieved in our solution.In the sequel, we analyze the confidentiality of the stored files in the presence of curious G and S.

数据机密性:正如第 2 节所解释的，文件大小和用户访问模式的知识对于 G 和 S 运营他们的业务是必不可少的。因此，隐藏这些信息在我们的解决方案中无法实现。在续集中，我们分析了存在好奇的 G 和 s 时存储文件的机密性。

Observe that both, the gateway G and the service provider S, only see the encrypted files, i.e., f ← enc(k, f).Therefore, if the underlying encryption scheme is secure, the contents of the files are protected against any eavesdropper unless the key k is leaked.Our server-assisted encryption scheme is an instance of a message-locked encryption scheme [14].As the key generation is oblivi-ous, G does not have any additional advantage when compared to a standard message-locked encryption adversary.As such, similar to MLE schemes, ClearBox achieves indistinguishability for unpre-dictable files.

请注意，网关 G 和服务提供商 S 都只看到加密文件，即 f ← enc(k，f)。因此，如果基础加密方案是安全的，除非密钥 k 被泄露，否则文件的内容将受到保护，不会被任何窃听者窃取。我们的服务器辅助加

密方案是消息锁定加密方案的一个实例[14]。由于密钥生成是不可靠的，与标准的消息锁定加密对手相比，G 没有任何额外的优势。因此，类似于最大似然估计方案，ClearBox 实现了不可口述文件的不可区分性。

Notice that a curious gateway G may guess H(fi) for popular (predictable) content fi and compute the corresponding key (as he knows the secret x) in order to identify if this fi is stored by some clients.A client who stores a low-entropy confidential file, can pro-tect against this attack, by appending a high-entropy string so that the file cannot be guessed anymore.However, the deduplication of the file is no longer possible.ClearBox offers a stronger protec-tion towards any other entity who does not know the secret value x, e.g., the service provider S. Recall that G rate-limits client re-quests for encryption keys, to slow down brute-force search attacks on predictable file contents (via the interface of gateway).

请注意，一个好奇的网关 G 可能会猜测流行(可预测)内容 fi 的 H(fi)，并计算相应的密钥(因为他知道秘密 x)，以便识别该 fi 是否由某些客户端存储。存储低熵机密文件的客户端可以通过附加一个高熵字符串来防止这种攻击，这样就不会再猜测该文件了。但是，文件的重复数据消除不再可能。ClearBox 为不知道秘密值 x 的任何其他实体(例如服务提供商 s)提供了更强的保护。回想一下，G rate-limited 客户端请求加密密钥，以减缓对可预测文件内容的暴力搜索攻击(通过网关的接口)。

4.DEPLOYMENT IN AMAZON S3 AND DROPBOX

4.亚马逊 S3 和 DROPBOX 的部署

In what follows, we evaluate a prototype implementation of ClearBox using Amazon S3 and Dropbox as a back-end storage.

在接下来的内容中，我们使用亚马逊 S3 和 Dropbox 作为后端存储来评估 ClearBox 的原型实现。

4.1 Implementation Setup

4.1 实施设置

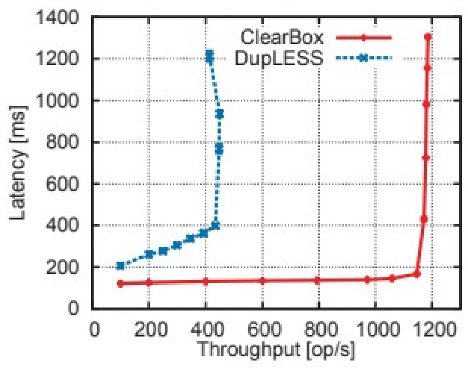
We implemented a prototype of ClearBox in Java.In our imple-mentation, we relied on SHA-256, the Java built-in random num-ber generator, and the JPBC library [7] (based on the PBC crypto-graphic library [5]) to implement BLS signatures.For a baseline comparison, we also implemented the server-based deduplication DupLESS of Bellare et al. [14] and integrated it with Amazon S3.Recall that, in DupLESS, clients directly interact with the cloud providers when storing/fetching their files.To generate keys, Dup-LESS uses a server-assisted oblivious protocol based on RSA blind signatures [14].Notice that we did not implement a plain (unen-crypted) cloud storage system since the performance of plain stor-age can be directly interpolated from the line rate (i.e., network capacity);where appropriate, we will discuss the performance of ClearBox relative to a plain cloud storage system.

我们用 Java 实现了一个 ClearBox 的原型。在我们的实现中，我们依赖于 SHA-256、Java 内置随机数字生成器和 JPBC 库[7](基于 PBC 加密图形库[5])来实现 BLS 签名。为了进行基线比较，我们还实施了 Bellare 等人[14]的基于服务器的重复数据消除 DupLESS，并将其与亚马逊 S3 公司集成。回想一下，在 DupLESS 中，客户端在存储/获取文件时直接与云提供商交互。为了生成密钥，Dup-LESS 使用基于 RSA 盲签名的服务器辅助遗忘协议[14]。请注意，我们没有实施普通(未加密)云存储系统，因为普通存储的性能可以直接根据线路速率(即网络容量)进行插值；在适当的地方，我们将讨论 ClearBox 相对于普通云存储系统的性能。

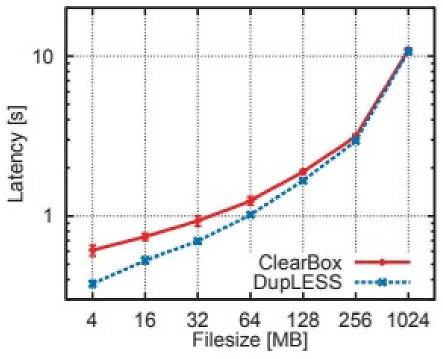
We deployed our implementations on two dual 6-core Intel Xeon E5-2640 clocked at 2.50GHz with 32GB of RAM, and 100Mbps network interface card.The ClearBox gateway, and the assisting server of DupLESS were running on one dual 6-core Xeon E5- 2640 machine, whereas the clients were co-located on the second dual 6-core Xeon E5-2640 machine;this ensures a fair compari-son between ClearBox and DupLESS.To emulate a realistic Wide Area Network (WAN), we relied on NetEm [38] to shape all traffic exchanged on the networking interfaces following a Pareto distribu-

我们在两个双 6 核英特尔至强 E5-2640 上部署了我们的实施，时钟频率为 2.50 千兆赫，内存为 32GB，网络接口卡为 100 兆位/秒。ClearBox 网关和 DupLESS 的辅助服务器运行在一台双 6 核至强 E5- 2640 机器上，而客户端位于第二台双 6 核至强 E5-2640 机器上；这确保了 ClearBox 和 DupLESS 之间的公平比较。为了模拟真实的广域网，我们依靠网络模型[38]按照帕累托分布来塑造网络接口上交换的所有流量

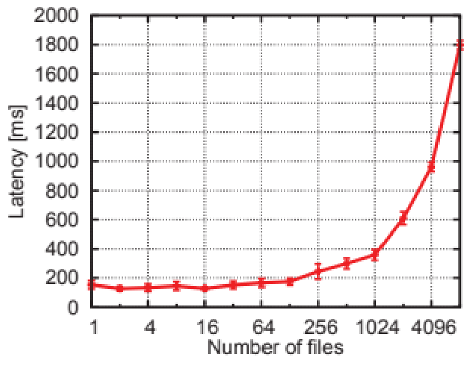
895

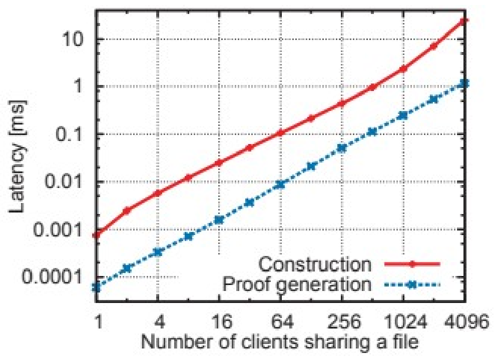


(a) Key generation performance at G



(b) Overhead of key generation on clients



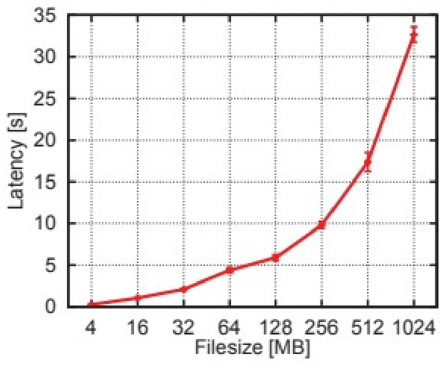


(d) CARDIAC construction and proof gener-ation

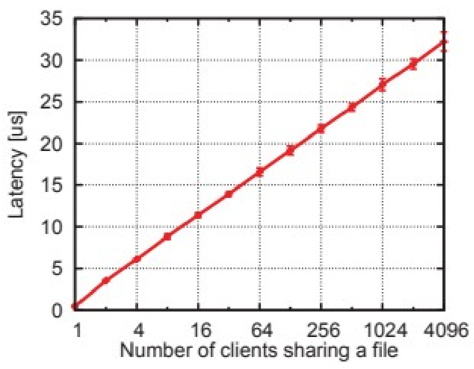
心脏结构和证据生成

(e) Latency in CARDIAC w.r.t. number of published files.

(e)CHARIC 中的延迟，已发布文件的数量。



(c) Construction of PoW



(f) CARDIAC verification by clients

客户的心脏检查

Figure 4: Performance evaluation of the building blocks used in ClearBox with respect to a number of parameters.

图 ClearBox 中使用的构建块在许多参数方面的性能评估。

tion with a mean of 20 ms and a variance of 4 ms (which emulates the packet delay variance of WANs [24]).

平均值为 20 毫秒，方差为 4 毫秒(模拟广域网的分组延迟方差[24])。

Our implementation interfaces with both Amazon S3 and Drop-box (respectively), which are used to store the user files.To acquire Bitcoin block hashes, our clients invoke an HTTP request to a get-blockhash tool offered by the Bitcoin block explorer[2].In our setup, each client invokes an operation in a closed loop, i.e., a client may have at most one pending operation.We spawned multiple threads on G's machine—each thread corresponding to a unique worker handling requests/ bills of a given client.We bounded the maximum number of threads that can be spawned in parallel to 100.In our implementation, the gateway and clients store the metadata information associated to each file in a local MySQL database.

我们的实现分别与亚马逊 S3 和 Drop-box 接口，用于存储用户文件。为了获取比特币块哈希，我们的客户端向比特币块浏览器提供的 get-blockhash 工具调用 HTTP 请求[2]。在我们的设置中，每个客户端在一个闭环中调用一个操作，即一个客户端最多只能有一个挂起的操作。我们在 G 的机器上产生了多个线程——每个线程对应于一个处理给定客户端的请求/账单的唯一工作者。我们将可以并行产生的最大线程数限制在 100 个。在我们的实现中，网关和客户端将与每个文件相关的元数据信息存储在本地 MySQL 数据库中。

The gateway leverages the caching feature of MySQL in or-der to reduce I/O costs.Recall that the Query Caching engine of MySQL [4] maps the text of the SELECT queries to their results.For each file, the gateway stores FID, the size of each file, and the IDs of the clients sharing the file.

该网关利用 MySQL 的缓存功能来降低输入/输出成本。回想一下，MySQL [4]的查询缓存引擎将选择查询的文本映射到它们的结果。对于每个文件，网关存储 FID、每个文件的大小以及共享该文件的客户端的标识。

Each data point in our plots is averaged over 10 independent measurements;where appropriate, we include the corresponding 95% confidence intervals.To accurately measure (micro-)latencies, we made use of the benchmarking tool due to Boyer [19].

我们的图中的每个数据点在 10 次独立测量中取平均值；在适当的情况下，我们包括相应的 95%置信区间。为了准确测量(微)延迟，我们利用了 Boyer [19]提出的基准工具。

4.2 Performance Evaluation

4.2 绩效评估

Before evaluating the overall performance of ClearBox, we start by analyzing the performance of the building blocks with respect to a number of parameters.Unless otherwise specified, we rely in our evaluation on the default parameters listed in Table 2.

在评估 ClearBox 的整体性能之前，我们首先从分析构建块在许多参数方面的性能开始。除非另有说明，否则我们的评估依赖于表 2 中列出的默认参数。

For example, the hash of Bitcoin block 'X' can be ac-

例如，比特币块“X”的散列可以是 ac-

quired by invoking https://blockexplorer.com/q/ getblockhash/X.

通过调用 https://blockexplorer.com/q/ getblockhash/x 获得

Gateway-assisted key generation: In Figure 4(a), we evaluate the overhead incurred by the oblivious key generation module (cf. Section 3.2.3) on the gateway.Here, we require that the gateway handles key generation requests back to back;we then gradually increase the number of requests in the system (until the through-put is saturated) and measure the associated latency.Our results show that our scheme incurs almost 125 ms latency per client key generation request on the gateway and attains a maximum through-put of 1185 operations per second;this considerably improves the maximum throughput of key generation of DupLESS (449 oper-ations per second).This is mainly due to the fact that BLS sig-natures are considerably faster to compute by the gateway when compared to RSA signatures.In contrast, as shown in Figure 4(b), BLS signatures are more expensive to verify by the clients than the RSA-variant employed in DupLESS.However, we argue that the overhead introduced by our scheme compared to DupLESS can be easily tolerated by clients, as the client effort is dominated by hashing the file;for instance, for 16 MB files, our proposal only in-curs an additional latency overhead of 213 ms on the clients when compared to DupLESS.

网关辅助密钥生成:在图 4(a)中，我们评估了不经意密钥生成模块(参见第 3.2.3 节)在网关上产生的开销。这里，我们要求网关背靠背地处理密钥生成请求；然后，我们逐渐增加系统中的请求数量(直到吞吐量达到饱和)，并测量相关的延迟。我们的结果表明，我们的方案在网关上的每一个客户端密钥生成请求都产生了几乎 125 ms 的延迟，并且达到了每秒 1185 个操作的最大吞吐量；这大大提高了双工密钥生成的最大吞吐量(每秒 449 次操作)。这主要是因为与 RSA 签名相比，网关计算 BLS 签名的速度要快得多。相比之下，如图 4(b)所示，客户端验证 BLS 签名比双工中使用的 RSA 变体更昂贵。然而，我们认为，与 DupLESS 相比，我们的方案引入的开销很容易被客户端容忍，因为客户端的工作主要是散列文件；例如，对于 16 MB 的文件，与双工相比，我们的建议仅在客户端上增加了 213 毫秒的延迟开销。

Proofs of ownership: Figure 4(c) depicts the overhead incurred by FID required to instantiate the PoW scheme of [27] with respect to the file size.As explained in Appendix B, the PoW scheme of [27]

所有权证明:图 4(c)描述了 FID 在文件大小方面实例化[27]的 PoW 方案所需的开销。如附录 B 中所解释的，[27]的功率方案

Parameter Default Value

参数默认值

Default file size 16 MB

默认文件大小 16 MB

RSA modulus size 2048 bit

RSA 模数大小 2048 位

|CFID | 100

|CFID | 100

Num.of PoW challenges 50

民数记 PoW 挑战 50

Elliptic Curve (BLS) PBC Library Curve F

椭圆曲线(BLS) PBC 图书馆曲线

Table 2: Default parameters used in evaluation.

表 2:评估中使用的默认参数。

896

reduces the costs of verifying PoW by encoding the original file in a bounded size buffer (64 MB in our case).Our results show that the computation of FID results in a tolerable overhead on both the gateway and the clients.For example, the computation of FID requires 1.07 seconds for a file with size 16 MB.CARDIAC: In Figure 4(d), we evaluate the overhead incurred by the construction of the accumulators and the proof generation in CARDIAC with respect to |CFID |, i.e., the number of clients sub-scribed to the same file f. Our results show that the latency in-curred by the construction of CARDIAC can be easily tolerated by G;for instance, the construction of an accumulator for 1000 users requires 2.34 ms. Clearly, this overhead increases as the number of users sharing the same file increases.Notice that once the entire Merkle tree is constructed, proof generation can be performed con-siderably faster than the construction of the accumulator.In this case, G only needs to traverse the tree, and record the sibling paths for all members of the accumulator.

通过在有限大小的缓冲区(在我们的例子中为 64 MB)中编码原始文件，降低了验证 PoW 的成本。我们的结果表明，FID 的计算在网关和客户端上都产生了可容忍的开销。例如，对于大小为 16 MB 的文件，FID 的计算需要 1.07 秒。心脏:在图 4(d)中，我们根据|CFID |评估了心脏中累加器和证明生成的构造所引起的开销，即同一文件 f 中描述的客户端数量。我们的结果表明，心脏的构造所产生的延迟很容易被 G 容忍；例如，为 1000 个用户构建累加器需要 2.34 毫秒。显然，这种开销会随着共享同一文件的用户数量的增加而增加。请注意，一旦构建了整个 Merkle 树，证明生成的执行速度将大大快于累加器的构建速度。在这种情况下，G 只需要遍历树，并记录累加器所有成员的兄弟路径。

In Figure 4(e), we evaluate the overhead incurred on G by the proof generation in CARDIAC with respect to the files selected for attestation at the end of each epoch.Our results show that this la-tency is around 150 ms when less than 100 files are selected, since the accumulators of these files can be handled in parallel by sep-arate threads in our pool.When the number of selected files in-creases beyond our pool threshold (i.e., 100), the latency of proof generation increases e.g., to reach almost 1 second for 4000 files.

在图 4(e)中，我们评估了在每个时期结束时，针对被选择用于证明的文件，在心动图中的证明生成在 G 上产生的开销。我们的结果表明，当选择的文件少于 100 个时，这个时间大约为 150 毫秒，因为这些文件的累加器可以由我们池中的独立线程并行处理。当所选文件的数量增加超过我们的池阈值(即 100 个)时，证据生成的延迟会增加，例如，对于 4000 个文件来说，达到几乎 1 秒。

In Figure 4(f), we evaluate the overhead of the proof verification by the clients.Given that the verification only requires log |CFID | hashes, this operation only incurs marginal overhead on the clients.For example, the verification of membership and cardinality in CARDIAC when |CFID | = 1000 only requires 27 μs. ClearBox: In Figure 5(a), we evaluate the latency witnessed by users in the PUT operation with respect to the file size.Our results show that ClearBox achieves comparable performance than Dup-LESS over S3.For example, when uploading 16 MB files on Ama-zon S3, DupLESS incurs a latency of 4.71 seconds, while ClearBox requires 6.33 seconds.The additional overhead in ClearBox mainly originates from the computation of FID by the users (cf. Fig-ure 4(c));the latency is mainly dominated by the upload of the file to Amazon.

在图 4(f)中，我们评估了客户验证的开销。假设验证只需要日志|CFID |哈希，这个操作只会给客户端带来少量开销。例如，当|CFID | = 1000 时，在心动图中验证成员资格和基数只需要 27μs。ClearBox:在图 5(a)中，我们评估了用户在 PUT 操作中看到的与文件大小相关的延迟。我们的结果表明，在 S3 上，ClearBox 实现了比 Dup-LESS 相当的性能。例如，在 Ama-zon S3 上传 16 MB 文件时，DupLESS 的延迟为 4.71 秒，而 ClearBox 需要 6.33 秒。ClearBox 中的额外开销主要来自用户对 FID 的计算(参见图 4(c))；延迟主要由上传到亚马逊的文件决定。

In contrast, when users want to upload a file which is already stored on the cloud, ClearBox results in faster upload performance than DupLESS, since users are no longer required to upload the file, but have to execute the PoW protocol with G—which incurs negligible latency when compared to the upload of modest-sized files.Recall that this comes at the expense of additional load on G;in contrast, the server in DupLESS bears no load when users upload/download files.

相比之下，当用户想要上传已经存储在云上的文件时，ClearBox 的上传性能比 DupLESS 更快，因为用户不再需要上传文件，而是必须执行带有 G 的 PoW 协议——与上传中等大小的文件相比，这种协议的延迟可以忽略不计。回想一下，这是以 G 的额外负载为代价的；相比之下，当用户上传/下载文件时，DupLESS 中的服务器不承担任何负载。

When using Dropbox, recall that clients upload the file directly to G, which in turn uploads it to its Dropbox account (since Dropbox does not provide URL commands for file creation).Although this process requires less communication rounds between the clients and G (to acquire the signed URL), our results show that the la-tency incurred when uploading un-deduplicated files in ClearBox using Dropbox is marginally larger than its Amazon S3 counter-part;we believe that this discrepancy is due to the lower upload bandwidth between our clients and G when compared to Amazon S3.Notice that the performance of uploading deduplicated files in ClearBox does not depend on the back-end cloud provider and is therefore identical for both our Amazon S3 and Dropbox imple-mentations.

当使用 Dropbox 时，回想一下客户端将文件直接上传到 G，G 又将文件上传到其 Dropbox 帐户(因为 Dropbox 不为文件创建提供 URL 命令)。虽然这个过程需要客户端和 G 之间更少的通信轮次(以获取签名的网址)，但我们的结果显示，使用 Dropbox 在 ClearBox 中上传未消除重复数据的文件时产生的延迟略大于其亚马逊 S3 计数器部分；我们认为，这种差异是由于与亚马逊 S3 相比，我们的客户端和 G 之间的上传带宽较低。请注意，在 ClearBox 中上传已消除重复数据的文件的性能不依赖于后端云提供商，因此对于我们的亚马逊 S3 和 Dropbox 实施来说是相同的。

In Figure 5(b), we evaluate the latency witnessed by users in the GET operation in ClearBox.Our results show that GET operation

在图 5(b)中，我们评估了用户在 ClearBox 的 GET 操作中看到的延迟。我们的结果表明，GET 操作

incurs comparable latencies in both ClearBox and DupLESS.Re-call that in ClearBox, clients have to first contact G and acquire the timed GET URL to download resources.Given that the la-tency of the GET operation is dominated by the download speed, the overhead incurred by this additional communication round is marginal.Notice that the latency exhibited by ClearBox users is tolerable compared to that witnessed in a plain cloud storage sys-tem.For instance, assuming a 100 Mbps line rate, the download of a 32MB file in plain clouds requests almost 3 seconds;on the other hand, downloading this file in ClearBox incurs a latency of 10 seconds.

在 ClearBox 和 DupLESS 中都产生了相当的延迟。在 ClearBox 中，客户端必须首先联系 G 并获取定时获取网址来下载资源。考虑到 GET 操作的效率是由下载速度决定的，这一轮额外的通信产生的开销是微不足道的。请注意，与普通云存储系统相比，ClearBox 用户表现出的延迟是可以容忍的。例如，假设线路速率为 100 Mbps，在普通云中下载一个 32MB 的文件几乎需要 3 秒钟；另一方面，在 ClearBox 中下载这个文件会导致 10 秒的延迟。

In Figure 5(c), we evaluate the latency incurred on the gateway in ClearBox with respect to the achieved throughput.Here, we as-sume that 50% of the requests handled by G correspond to PUT requests, while the remaining 50% are GET requests.We further assume that 50% of the upload requests correspond to files which are already stored at S. When uploading (un-deduplicated) files to S, we do not measure the overhead incurred on G when verifying FID since this verification is asynchronous and can be done by G at a later point in time.We further simulate a large download bandwidth at G by emulating client requests from a local socket.Our findings show that ClearBox achieves a maximum throughput of approximately 2138 operations per second when integrated with Amazon S3.This shows that our solution scales to a large number of users in the system.When interfacing with Dropbox, the perfor-mance of ClearBox deteriorates since the load on G considerably increases in this case;the maximum throughput exhibited by our scheme decreases to almost 289 operations per second.

在图 5(c)中，我们根据获得的吞吐量评估了 ClearBox 中网关的延迟。这里，我们假设 G 处理的 50%的请求对应于 PUT 请求，而剩下的 50%是 GET 请求。我们进一步假设 50%的上传请求对应于已经存储在

S 的文件。当将(未消除重复数据的)文件上传到 S 时，我们不会在验证 FID 时测量 G 产生的开销，因为这种验证是异步的，可以由 G 在稍后的时间点完成。我们通过模拟来自本地套接字的客户端请求，进一步模拟了 G 的大下载带宽。我们的发现表明，当与亚马逊 S3 集成时，ClearBox 实现了大约每秒 2138 次操作的最大吞吐量。这表明我们的解决方案可以扩展到系统中的大量用户。当与 Dropbox 接口时，ClearBox 的性能会下降，因为在这种情况下 G 上的负载会大大增加；我们的方案显示的最大吞吐量下降到几乎每秒 289 次操作。

5.RELATED WORK

5.相关著作

Data deduplication in cloud storage systems has acquired con-siderable attention in the literature.

云存储系统中的重复数据删除在文献中获得了相当多的关注。

In [28], Harnik et al. describe a number of threats posed by client-side data deduplication, in which an adversary can learn if a file is already stored in a particular cloud by guessing the hashes of predictable messages.This leakage can be countered using Proofs of Ownership schemes (PoW) [23, 27], which enable a client to prove it possesses the file in its entirety.PoW are inspired by Proofs of Retrievability and Data Possession (POR/PDP) schemes [11,40], with the difference that PoW do not have a pre-processing step at setup time.Halevi et al. [27] propose a PoW construct based on Merkle trees which incurs low overhead on the server in construct-ing and verifying PoW.Xu et al. [44] build upon the PoW of [27] to construct a PoW scheme that supports client-side deduplication in a bounded leakage setting.Di Pietro and Sorniotti [23] propose a PoW scheme which reduces the communication complexity of [27] at the expense of additional server computational overhead.Blasco et al. [16] propose a PoW based on Bloom filters which further reduces the server-side overhead of [23].

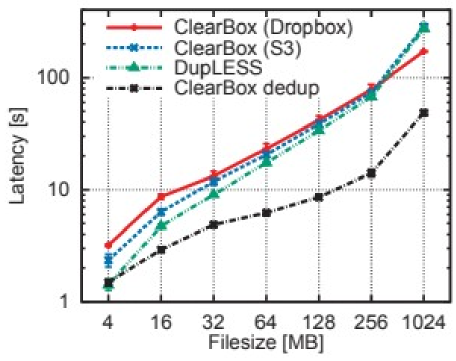
在[28]中，Harnik 等人描述了客户端重复数据消除带来的许多威胁，其中对手可以通过猜测可预测消息的散列来了解文件是否已经存储在特定云中。可以使用所有权证明方案(PoW) [23，27]来应对这种泄露，该方案使客户能够证明其拥有整个文件。PoW 的灵感来自可检索性和数据拥有证明(POR/PDP)方案[11，40]，不同之处在于 PoW 在建立时没有预处理步骤。Halevi 等人[27]提出了一种基于 Merkle 树的 PoW 构造，在构造和验证 PoW 时，服务器的开销很低。Xu 等人[44]在[27]的 PoW 的基础上构建了一个 PoW 方案，该方案支持在有界泄漏设置下的客户端重复数据消除。Di Pietro 和 Sorniotti [23]提出了一种 PoW 方案，以额外的服务器计算开销为代价，降低了[27]的通信复杂度。布拉斯科等人[16]提出了一种基于布隆过滤器的 PoW，进一步降低了[23]的服务器端开销。

Douceur et al. [25] introduced the notion of convergent encryp-tion, a type of deterministic encryption in which a message is en-crypted using a key derived from the plaintext itself.Convergent encryption is not semantically secure [15] and only offers confi-dentiality for messages whose content is unpredictable.Bellare et al. [31] proposed DupLESS, a server-aided encryption to perform data deduplication scheme;here, the encryption key is obliviously computed based on the hash of the file and the private key of the as-sisting server.In [42], Stanek et al. propose an encryption scheme which guarantees semantic security for unpopular data and weaker security (using convergent encryption) for popular files.In [41], For comparison purposes, we did not include in our measurements the overhead introduced by the uploading of files by G onto S;this process can be executed at a later point in time by G.

Douceur 等人[25]引入了收敛加密的概念，这是一种确定性加密，其中使用从明文本身导出的密钥对消息进行加密。聚合加密在语义上是不安全的[15]，并且只为内容不可预测的消息提供一致性。Bellare 等

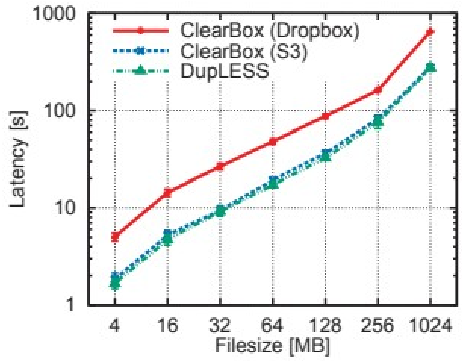
人[31]提出了 DupLESS，一种服务器辅助加密来执行重复数据删除方案；在这里，加密密钥是基于文件的散列和辅助服务器的私钥来明确计算的。在[42]中，Stanek 等人提出了一种加密方案，该方案保证了不受欢迎的数据的语义安全性和受欢迎文件的较弱安全性(使用收敛加密)。在[41]中，为了比较的目的，我们没有在我们的测量中包括由 G 上传文件到 S 而引入的开销；这个过程可以由 g 在稍后的时间点执行

897



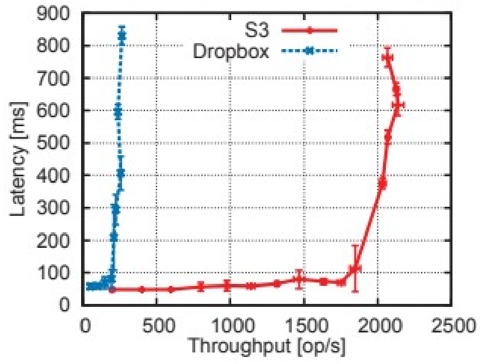
(a) PUT Performance

投入产出绩效



(b) GET Performance

获得绩效



(c) Latency vs. throughput

延迟与吞吐量

Figure 5: Performance evaluation of ClearBox using Amazon S3 and Dropbox as back-end cloud storage.

图 ClearBox 使用亚马逊 S3 和 Dropbox 作为后端云存储的性能评估。

Soriente et al. proposed a solution which distributively enforces shared ownership in agnostic clouds.This solution can be used in conjunction with ClearBox to enable users to distributively manage the access control of their deduplicated files.

Soriente 等人提出了一种解决方案，在不可知云中分布式地实施共享所有权。该解决方案可以与 ClearBox 结合使用，使用户能够分布式管理其已消除重复数据的文件的访问控制。

Several proposals for accumulating hidden sets exist such as the original RSA-based construction [12], which has been extended to dynamic accumulators [21], and non-membership proofs [32].There exist as well constructions based on bilinear groups [22, 39] and on hash-functions [20, 33].A related concept are zero-knowledge sets [30, 37].These structures provide proofs of set membership.Notice, however, that cryptographic accumulators do not typically provide information about the accumulated set, such as content or cardinality.

有几个积累隐藏集的建议，如最初的基于 RSA 的构造[12]，已扩展到动态累加器[21]，以及非成员证明[32]。也存在基于双线性群[22，39]和基于散列函数[20，33]的构造。一个相关的概念是零知识集[30，37]。这些结构提供了集合成员的证明。但是，请注意，加密累加器通常不提供有关累加集的信息，如内容或基数。

Our attacker model shares similarities with the one considered in [43], where the cloud provider is assumed to be economically ra-tional.This scheme relies on an hourglass function—to verify that the cloud provider stores the files in encrypted form—which im-poses significant constraints on a rational cloud provider that tries to apply the hourglass functions on demand.

我们的攻击者模型与[43]中考虑的模型有相似之处，在该模型中，云提供商被认为在经济上是跨国的。该方案依赖沙漏函数——验证云提供商以加密形式存储文件——这对试图按需应用沙漏函数的 rational cloud provider 造成了重大限制。

6.CONCLUSION

6.结论

In this paper, we proposed ClearBox, which enables a cloud provider to transparently attest to its clients the deduplication pat-terns of their stored data.ClearBox additionally enforces fine-grained access control over deduplicated files, supports data con-fidentiality, and resists against malicious users.Evaluation results derived from a prototype implementation of ClearBox show that our proposal scales well with the number of users and files in the system.

在本文中，我们提出了 ClearBox，它使云提供商能够透明地向其客户证明其存储数据的重复数据消除模式。ClearBox 还对经过重复数据消除的文件实施细粒度的访问控制，支持数据真实性，并抵御恶意用户。从 ClearBox 的原型实现中得到的评估结果表明，我们的建议可以很好地适应系统中用户和文件的数量。

As far as we are aware, ClearBox is the first complete system which enables users to verify the storage savings exhibited by their data.We argue that ClearBox motivates a novel cloud pricing model which promises a fairer allocation of storage costs amongst users—without compromising data confidentiality nor system per-formance.We believe that such a model provides strong incentives for users to store popular data in the cloud (since popular data will be cheaper to store) and discourages the upload of personal and unique content.As a by-product, the popularity of files addition-ally gives an indication to cloud users on their level of privacy in the cloud;for example, the user can verify that his private files are not deduplicated—and thus have not been leaked.

据我们所知，ClearBox 是第一个完整的系统，它使用户能够验证他们的数据所展示的存储节省。我们认为，ClearBox 激发了一种新的云定价模式，这种模式承诺在用户之间更公平地分配存储成本，而不会损害数据机密性或系统性能。我们认为，这种模式为用户在云中存储流行数据提供了强大的激励(因为存储流行数据会更便宜)，并阻止个人和独特内容的上传。作为一个副产品，文件添加的流行向云用户表明了他们在云中的隐私级别；例如，用户可以验证他的私有文件没有经过重复数据消除，因此没有被泄露。

Acknowledgements

承认

The authors would like to thank Nikos Triandopoulos and the anony-mous reviewers for their valuable feedback and comments.This work was partly supported by the TREDISEC project (G.A. no 644412), funded by the European Union (EU) under the Informa-

作者要感谢尼科斯·特里亚多普洛斯和匿名审稿人的宝贵反馈和意见。这项工作得到了欧洲联盟(欧盟)资助的特雷迪斯项目(政府编号 644412)的部分支持

tion and Communication Technologies (ICT) theme of the Horizon 2020 (H2020) research and innovation programme.

地平线 2020 (H2020)研究和创新计划的信息和通信技术主题。

7.REFERENCES

7.参考

[1] Amazon S3 Pricing.

[1]亚马逊 S3 定价。

http://aws.amazon.com/s3/pricing/.

http://aws.amazon.com/s3/pricing/.

[2] Bitcoin real-time stats and tools.http://blockexplorer.com/q.

[2]比特币实时统计和工具。http://blockexplorer.com/q.

[3] Google Cloud Storage.

[3]谷歌云存储。

https://cloud.google.com/storage/.

https://cloud.google.com/storage/.

[4] The MySQL Query Cache.http://dev.mysql.com/ doc/refman/5.1/en/query-cache.html.

MySQL 查询缓存。http://dev.mysql.com/文件/ref man/5.1/en/query-cache . html。

[5] PBC Library.http://crypto.stanford.edu/pbc/, 2007.

[5] PBC 图书馆。http://crypto.stanford.edu/pbc/, 2007。

[6] Cloud Market Will More Than Triple by 2014, Reaching $150 Billion.

[6]到 2014 年，云市场将增长两倍以上，达到 1500 亿美元。

http://www.msptoday.com/topics/msp-today/articles/364312-cloud-market-will-more-than-triple-2014-reaching.htm, 2013.

http://www . MSP today . com/topics/MSP-today/articles/364312-cloud-market-will-over-triple-2014-reach . htm，2013。

[7] JPBC:Java Pairing-Based Cryptography Library.http://gas.dia.unisa.it/projects/jpbc/#. U3HBFfna5cY, 2013.

[7] JPBC:基于 Java 配对的密码库。http://gas.dia.unisa.it/projects/jpbc/#. U3HBFfna5cY，2013 年。

[8] Bitcoin as a public source of randomness.https://docs.google.com/presentation/d/ 1VWHm4Moza2znhXSOJ8FacfNK2B_ vxnfbdZgC5EpeXFE/view?pli=1#slide=id.g3934beb89_034, 2014.

[8]比特币作为随机性的公共来源。https://docs.google.com/presentation/d/ 1vwhm4moza znhxsoj8 facfnk2b _ vxnfbdzgc5 epexfe/view？ pli = 1 #幻灯片=id。g3934beb89_034，2014。

[9] These are the cheapest cloud storage providers right now.http://qz.com/256824/these-are-the-cheapest-cloud-storage-providers-right-now/, 2014.

[9]这些是目前最便宜的云存储提供商。http://qz . com/256824/这些是目前最便宜的云存储提供商，2014 年。

[10] ARMKNECHT, F., BOHLI, J., KARAME, G. O., LIU, Z., AND REUTER, C. A. Outsourced proofs of retrievability.In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014 (2014), pp. 831-843.

[10] ARMKNECHT，f .，BOHLI，j .，KARAME，G. O .，LIU，z .，和 REUTER，C. A .可检索性的外包证明。2014 年 11 月 3 日至 7 日在美国亚利桑那州斯科茨代尔举行的 2014 年计算机和通信安全 ACM SIGSAC 会议记录(2014 年)，第 831-843 页。

[11] ATENIESE, G., BURNS, R. C., CURTMOLA, R., HERRING, J., KISSNER, L., PETERSON, Z. N. J., AND SONG, D. X. Provable data possession at untrusted stores.In ACM Conference on Computer and Communications Security (2007), pp. 598-609.

[11] ATENIESE，g，BURNS，R.
C .，CURTMOLA，r .，HERRING，j .，KISSNER，l .，PETERSON，Z. N. J .，和 SONG，D. X .在不可信商店可证明的数据占有。计算机与通信安全会议(2007)，第 598-609 页。

[12] BARIC, N., AND PFITZMANN, B. Collision-free accumulators and fail-stop signature schemes without trees.In EUROCRYPT (1997), W. Fumy, Ed., vol. 1233 of Lecture Notes in Computer Science, Springer, pp. 480-494.

[12]无冲突累加器和无树故障停止签名方案。在 EUROCRYPT (1997)，W. Fumy，Ed。《计算机科学讲义》第 1233 卷，斯普林格出版社，第 480-494 页。

898

898

[13] BELLARE, M., AND KEELVEEDHI, S. Interactive message-locked encryption and secure deduplication.In Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings (2015), J. Katz, Ed., vol. 9020 of Lecture Notes in Computer Science, Springer, pp. 516-538.

[13]交互消息锁定加密和安全重复数据删除。在公钥密码学中- PKC 2015 -第 18 届 IACR 公钥密码学实践和理论国际会议，美国马里兰州盖瑟斯堡，2015 年 3 月 30 日-4 月 1 日，会议记录(2015)，j .卡茨，编辑。《计算机科学讲义》第 9020 卷，斯普林格出版社，第 516-538 页。

[14] BELLARE, M., KEELVEEDHI, S., AND RISTENPART, T. DupLESS: Server-aided encryption for deduplicated storage.In Proceedings of the 22Nd USENIX Conference on Security (Berkeley, CA, USA, 2013), SEC'13, USENIX Association, pp. 179-194.

[14]贝拉尔，m .，KEELVEEDHI，s .，AND RISTENPART，T. DupLESS:用于重复数据消除存储的服务器辅助加密。《第 22 届 USENIX 安全会议论文集》(美国加州柏克莱，2013 年)，
SEC'13，USENIX 协会，第 179-194 页。

[15] BELLARE, M., KEELVEEDHI, S., AND RISTENPART, T. Message-locked encryption and secure deduplication.In Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013.Proceedings (2013), T. Johansson and P. Q. Nguyen, Eds., vol. 7881 of Lecture Notes in Computer Science, Springer, pp. 296-312.

[15]信息锁定加密和安全重复数据删除。2013 年 5 月 26 日至 30 日在希腊雅典举行的第 32 届密码技术理论和应用年度国际会议上发表的《密码学进展——EUROCRYPT 2013》。会议记录(2013 年)，t .约翰逊和 P. Q. Nguyen 编辑。《计算机科学讲义》第 7881 卷，斯普林格出版社，第 296-312 页。

[16] BLASCO, J., DI PIETRO, R., ORFILA, A., AND SORNIOTTI, A. A tunable proof of ownership scheme for deduplication using bloom filters.In Communications and Network Security (CNS), 2014 IEEE Conference on (Oct 2014), pp. 481-489.

[16] BLASCO，j .，DI PIETRO，r .，ORFILA，A .，和 SORNIOTTI，A .一种使用 bloom 过滤器的重复数据消除的所有权方案的可调证明。在通信和网络安全(CNS)中，2014 年 IEEE 会议(2014 年 10 月)，第 481-489 页。

[17] BOLDYREVA, A. Efficient threshold signature, multisignature and blind signature schemes based on the gap-diffie-hellman-group signature scheme.

[17] BOLDYREVA，a .基于 gap-diffie-hellman-群签名方案的高效门限签名、多重签名和盲签名方案。

[18] BONEH, D., LYNN, B., AND SHACHAM, H. Short signatures from the weil pairing.J. Cryptology 17, 4 (2004), 297-319.

[18]韦尔配对的短签名。《密码学》17，4 (2004)，297-319。

[19] BRENT BOYER.Robust Java benchmarking.http: //www.ibm.com/developerworks/library/j-benchmark2/j\-benchmark2\-pdf.pdf.

[19] BRENT BOYER。健壮的 Java 基准测试。http://www . IBM . com/developer works/library/j-bench mark2/j \-bench mark2 \-pdf . pdf。

[20] BULDAS, A., LAUD, P., AND LIPMAA, H. Eliminating counterevidence with applications to accountable certificate management.Journal of Computer Security 10, 3 (2002), 273-296.

[20]布尔达斯，美国，劳德，p，和利普马，h .用应用程序消除反证据，以负责任的证书管理。计算机安全杂志 10，3 (2002)，273-296。

[21] CAMENISCH, J., AND LYSYANSKAYA, A. Dynamic accumulators and application to efficient revocation of anonymous credentials.In Advances in Cryptology - CRYPTO 2002 (2002), Springer, pp. 61-76.

动态累加器及其在有效撤销匿名证书中的应用。《密码学进展-密码 2002》(2002)，斯普林格，第 61-76 页。

[22] DAMGÅRD, I., AND TRIANDOPOULOS, N. Supporting non-membership proofs with bilinear-map accumulators.IACR Cryptology ePrint Archive 2008 (2008), 538.

[22]用双线性映射累加器支持非成员证明。IACR 密码学电子档案 2008 (2008)，538。

[23] DI PIETRO, R., AND SORNIOTTI, A. Boosting efficiency and security in proof of ownership for deduplication.In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (New York, NY, USA, 2012), ASIACCS '12, ACM, pp. 81-82.

[23]迪·皮埃特罗和索尼奥蒂在重复数据消除的所有权证明中提高了效率和安全性。《第七届美国计算机学会信息、计算机和通信安全研讨会论文集》(美国纽约州纽约市，2012 年)，《亚洲计算机安全会议》第 12 期，美国计算机学会，第 81-82 页。

[24] DOBRE, D., KARAME, G., LI, W., MAJUNTKE, M., SURI, N., AND VUKOLIC , M. Powerstore: Proofs of writing for efficient and robust storage.In Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38;Communications Security (New York, NY, USA, 2013), CCS '13, ACM, pp. 285-298.

[24] DOBRE，d .，KARAME，g .，LI，w .，MAJUNTKE，m .，SURI，n .，和 VUKOLIC，M. Powerstore:高效和稳健存储的写作证明。在 2013 年美国计算机学会计算机会议录& # 38；《通信安全》(美国纽约州纽约市，2013 年)，《通信安全通讯》第 13 期，美国计算机学会，第 285-298 页。

[25] DOUCEUR, J. R., ADYA, A., BOLOSKY, W. J., SIMON, D., AND THEIMER, M. Reclaiming space from duplicate files in a serverless distributed file system.In ICDCS (2002), pp. 617-624.

[25]无服务器分布式文件系统中从重复文件中回收空间。在国际发展合作中心(2002 年)，第 617-624 页。

[26] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems.In Proceedings on Advances in cryptology—CRYPTO '86 (London, UK, UK, 1987), Springer-Verlag, pp. 186-194.

[26] FIAT，a .，AND SHAMIR，a .如何证明自己:识别和签名问题的实用解决方案。《密码学进展论文集——CRYPTO’86》(伦敦，英国，英国，1987)，斯普林格-弗拉格，第 186-194 页。

[27] HALEVI, S., HARNIK, D., PINKAS, B., AND SHULMAN-PELEG, A. Proofs of ownership in remote storage systems.In Proceedings of the 18th ACM Conference on Computer and Communications Security (New York, NY, USA, 2011), CCS '11, ACM, pp. 491-500.

[27]哈列维，s .，哈尼克，d .，平卡斯，b .，和舒尔曼-佩莱格，a .远程存储系统所有权的证明。在第 18 届计算机和通信安全会议记录(纽约，纽约，美国，2011 年)，CCS '11，ACM，第 491-500 页。

[28] HARNIK, D., PINKAS, B., AND SHULMAN-PELEG, A. Side channels in cloud services: Deduplication in cloud storage.IEEE Security & Privacy 8, 6 (2010), 40-47.

[28] HARNIK，d .，PINKAS，b .，和 SHULMAN-PELEG，a .云服务中的辅助通道:云存储中的重复数据消除。IEEE 安全与隐私 8，6 (2010)，40-47。

[29] KARAME, G. O., ANDROULAKI, E., AND CAPKUN, S. Double-spending fast payments in bitcoin.In Proceedings of the 2012 ACM conference on Computer and communications security (New York, NY, USA, 2012), CCS '12, ACM, pp. 906-917.

[29] KARAME，G. O .，ANDROULAKI，e .，AND CAPKUN，s .比特币双消费快速支付。《2012 年美国计算机学会计算机和通信安全会议记录》(美国纽约州纽约市，2012 年)，CCS '12，美国计算机学会，第 906-917 页。

[30] KATE, A., ZAVERUCHA, G. M., AND GOLDBERG, I. Constant-size commitments to polynomials and their applications.In Advances in Cryptology-ASIACRYPT 2010.Springer, 2010, pp. 177-194.

[30] KATE，a .，ZAVERUCHA，G. M .，和 GOLDBERG，I .多项式的常数大小承诺及其应用。密码学进展-亚洲密码 2010。斯普林格，2010 年，第 177-194 页。

[31] KEELVEEDHI, S., BELLARE, M., AND RISTENPART, T. DupLESS: Server-aided encryption for deduplicated storage.In Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13) (Washington, D.C., 2013), USENIX, pp. 179-194.

[31]keelevedhi，s .，BELLARE，m .，AND RISTENPART，T. DupLESS:服务器辅助加密用于已消除重复数据的存储。作为第 22 届 USENIX 安全研讨会(USENIX 安全 13)的一部分提交(华盛顿特区，2013 年)，USENIX，第 179-194 页。

[32] LI, J., LI, N., AND XUE, R. Universal accumulators with efficient nonmembership proofs.In Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings (2007), pp. 253-269.

具有有效非成员证明的泛累加器。应用密码学与网络安全，第五届国际会议，ACNS 2007，中国珠海，2007 年 6 月 5-8 日，会议录(2007)，第 253-269 页。

[33] LIPMAA, H. Secure accumulators from euclidean rings without trusted setup.In Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012.Proceedings (2012), pp. 224-240.

[33]利普马，h .在没有可信设置的情况下从欧几里得环中保护累加器。2012 年 6 月 26 日至 29 日，在 ACNS 举行的第十届应用密码学与网络安全国际会议上。《诉讼程序》(2012 年)，第 224-240 页。

[34] LIU, S., HUANG, X., FU, H., AND YANG, G. Understanding data characteristics and access patterns in a cloud storage system.In 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013, Delft, Netherlands, May 13-16, 2013 (2013), pp. 327-334.

[34]刘，s，黄，x，付，h，和杨，g .了解云存储系统中的数据特征和访问模式。在第十三届 IEEE/ACM 集群、云和网格计算国际研讨会上，CCGrid 2013，荷兰代尔夫特，2013 年 5 月 13-16 日(2013)，第 327-334 页。

[35] MEYER, D. T., AND BOLOSKY, W. J. A study of practical deduplication.In Proceedings of the 9th USENIX Conference on File and Stroage Technologies (Berkeley, CA, USA, 2011), FAST'11, USENIX Association, pp. 1-1.

35《实用重复数据删除研究》。文件和存储技术第九届 USENIX 会议记录(美国加州柏克莱，2011 年)，FAST'11，USENIX 协会，第 1-1 页。

[36] MEYER, D. T., AND BOLOSKY, W. J. A study of practical deduplication.Trans.Storage 7, 4 (Feb. 2012), 14:1-14:20.

36《实用重复数据删除研究》。跨。存储 7，4(2012 年 2 月)，14:1-14:20。

[37] MICALI, S., RABIN, M., AND KILIAN, J. Zero-knowledge sets.In Foundations of Computer Science, 2003.Proceedings.44th Annual IEEE Symposium on (2003), IEEE, pp. 80-91.

[37]零知识集。《计算机科学基础》，2003 年。诉讼程序。第 44 届 IEEE 年会(2003 年)，IEEE，第 80-91 页。

[38] NETEM.NetEm, the Linux Foundation.Website, 2009.Available online at

38 NETEM。Linux 基金会。网站，2009 年。在线提供，网址为

http://www.linuxfoundation.org/ collaborate/workgroups/networking/netem.

http://www.linuxfoundation.org/协作/工作组/网络/网络。

[39] NGUYEN, L. Accumulators from bilinear pairings and applications.In Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings (2005), pp. 275-292.

[39] NGUYEN，l .双线性对的累加器及其应用。2005 年 2 月 14 日至 18 日在美国加利福尼亚州旧金山举行的 2005 年 RSA 会议上的密码学家轨迹，会议录(2005 年)，第 275-292 页。

899

899

[40] SHACHAM, H., AND WATERS, B. Compact Proofs of Retrievability.In ASIACRYPT (2008), pp. 90-107.

40《可检索性的紧凑证明》。在 ASIACRYPT (2008)，第 90-107 页。

[41] SORIENTE, C., KARAME, G. O., RITZDORF, H., MARINOVIC, S., AND CAPKUN, S. Commune: Shared ownership in an agnostic cloud.In Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, June 1-3, 2015 (2015), pp. 39-50.

[41] SORIENTE，c .，KARAME，G. O .，RITZDORF，h .，MARINOVIC，s .，AND CAPKUN，S. Commune:不可知论云中的共享所有权。2015 年 6 月 1 日至 3 日在奥地利维也纳举行的第 20 届访问控制模型和技术学术研讨会会议录(2015 年)，第 39-50 页。

[42] STANEK, J., SORNIOTTI, A., ANDROULAKI, E., AND KENCL, L. A secure data deduplication scheme for cloud storage.In Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers (2014), pp. 99-118.

[42]j . STANEK，A . SORNIOTTI，e . androlaki，和 l . KENCL .一种用于云存储的安全重复数据删除方案。在金融密码学和数据安全-第 18 届国际会议，FC 2014，基督教堂，巴巴多斯，2014 年 3 月 3 日至 7 日，修订论文选集(2014 年)，第 99-118 页。

[43] VAN DIJK, M., JUELS, A., OPREA, A., RIVEST, R. L., STEFANOV, E., AND TRIANDOPOULOS, N. Hourglass schemes: How to prove that cloud files are encrypted.In Proceedings of the 2012 ACM Conference on Computer and Communications Security (New York, NY, USA, 2012), CCS '12, ACM, pp. 265-280.

[43] VAN DIJK，m .，JUELS，a .，OPREA，a .，RIVEST，R. L .，STEFANOV，e .，AND TRIANDOPOULOS，n .沙漏方案:如何证明云文件是加密的。在 2012 年美国计算机学会计算机和通信安全会议记录(美国纽约州纽约市，2012 年)中，CCS '12，美国计算机学会，第 265-280 页。

[44] XU, J., CHANG, E.-C., AND ZHOU, J. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage.In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (New York, NY, USA, 2013), ASIA CCS '13, ACM, pp. 195-206.

[44]徐，j，CHANG，e-c，AND ZHOU，j .云存储中加密数据的弱泄漏弹性客户端重复数据删除。在第八届美国计算机学会信息、计算机和通信安全专题讨论会(2013 年，美国纽约)会议录中，亚洲计算机安全会议第 13 期，美国计算机学会，第 195-206 页。

APPENDIX

附录

A.SECURITY ANALYSIS OF CARDIAC

A.心脏的安全性分析

In what follows, we analyze the security of CARDIAC.Let a set X be given and the accumulated digest for the set is δ ← Acc(X).Recall that δ = H(a

接下来，我们分析了 CARDIAC 的安全性。设一个集合 X，该集合的累计摘要为 δ ← Acc(X)。回想一下 δ = H(a

-,0,) (with H being a cryptographically se-cure hash function) acts as a commitment to the root a

-，0，)作为对根 a 的承诺(H 是一个加密的安全散列函数)

-,0 and the height

-，0 和高度

of a Merkle tree.For the purpose of our analysis, we dis-tinguish between two security goals: (i) proving membership using ProveM and VerifyM and (ii) proving an upper bound on |X| us-ing ProveC and VerifyC.As the security of Merkle trees is well understood with respect to the first goal [20], we focus in the sequel on analyzing the second goal.

一棵 Merkle 树。为了分析的目的，我们区分了两个安全目标:(I)使用 ProveM 和 VerifyM 证明成员资格，以及(ii)使用 ProveC 和 VerifyC 证明|X|的上限。由于 Merkle 树的安全性对于第一个目标[20]有很好的理解，因此我们在后续文章中重点分析第二个目标。

Since

因为

is verified (indirectly) using the length of the sibling path and assuming that the hash function is secure, CARDIAC ensures that the Merkle tree can encode at most 2elements.ProveC out-puts the sibling path of the last set element a0,|X|−1.The verifi-cation algorithm VerifyC requires to check the values of all open nodes in the sibling path.Recall that the open nodes are those nodes that depend only on the zero leaves and hence represent publicly known values.In fact, this verification step constitutes member-ship proofs for every single zero leaf a0,j for j = |X|,..., 2− 1.In turn, this ensures that at least 2−|X| leaves of the tree are zero.

使用兄弟路径的长度(间接)进行验证，并假设哈希函数是安全的，则 chemical 确保 Merkle 树最多可以编码 2 个元素。ProveC out-放入最后一个集合元素 a0，| X |-1 的兄弟路径。verify 算法 VerifyC 要求检查同级路径中所有打开节点的值。回想一下，开放节点是那些仅依赖于零叶的节点，因此代表了公开的

已知值。事实上，这个验证步骤构成了每个单个零叶 a0、j 的成员资格证明，对于 j = |X|，..., 2− 1.反过来，这确保了树的至少 2 | X |叶为零。

A direct consequence is that at most |X| leaves can be non-zero, giving an upper bound on X. Observe that the proof assumes that further leaves located on the left of the last non-zero element are not set to zero.This assumption can be safely made since G has no incentives to place more zero-leaves (since this means that less users are registered than claimed and would pay too little to G).

一个直接的结果是最多|X|个叶子可以是非零的，给出了 X 的一个上限。请注意，该证明假设位于最后一个非零元素左侧的其他叶子没有设置为零。这一假设可以放心地做出，因为 G 没有动机放置更多的零叶(因为这意味着注册的用户比声称的要少，并且支付给 G 的费用太少)。

Input: A file f of length M bit, broken into m = M 512 blocks of length 512 bit.Initialize positions:
• Compute bit-length
= min {20, log2 m }, being close to M, but at most 64 MB, i.e. 2blocks with 512 bits.• Initialize the buffer Buf with 2blocks of 512 bits.• Initialize a table ptr of m rows and 4 columns of
bits
which holds pointers into the buffer Buf.A temporary value IV will be initialized with SHA-256' IV .Reduction phase: For each i ∈ [m]:
1.Update IV := SHA-256 (IV ;File[i]), where File[i] de-notes the i-th block of the file.
Initialize ptr [i] = trunc4
-(IV) with IV truncated to 4
bits..2.For j =0:3, do
3.Block = Cyclic Shift of F ile[i] by j * 128 bits 4.XOR Block into location ptr [i][j] in Buf Mixing phase: Repeat 5 times:
1.For each block i ∈ [
], For j =0:3, do 2.Block = Cyclic Shift of Buf[i] by j * 128 bits 3.If i = ptr [i mod 2][j] then XOR Block into location
ptr [i mod 2][j] in Buf
Tree construction: The final buffer content is denoted by Buf(f).Construct a Merkle tree using the blocks of Buf(f) as the leaves.

输入:长度为 M 位的文件 f，分成长度为 512 位的 m = M 个 512 块。初始化位置:
计算位长
= min {20，log2 m }，接近 M，但最多 64 MB，即 2 个 512 位的块。用 2 个 512 位的块初始化缓冲区 Buf。初始化一个 m 行 4 列的表 ptr
位
它保存指向缓冲区 Buf 的指针。一个临时值四将被初始化为 SHA-256 的四。缩减阶段:对于每一个 i ∈ [m]:
1.更新四:= SHA-256(四；文件[i])，其中文件[i]记录文件的第一个块。
初始化 ptr [i] = trunc4
-(iv)iv 被截断为 4
位..2.对于 j =0:3，做
3.Block =文件[i]的循环移位 j * 128 位 4。在混合阶段将异或块放入位置 ptr [i][j]:重复 5 次:
1.对于每个块 I∞[
】，对于 j =0:3，做 2。block = Buf[I]的循环移位 j * 128 位 3。如果 i = ptr [i mod 2][j]，则异或块进

入位置
Buf 中的 ptr [i mod 2][j]
树结构:最终的缓冲区内容由 Buf(f)表示。使用 Buf(f)的块作为叶子来构建 Merkle 树。

Figure 6: Detailed PoW Protocol of [27].

图 6:详细的 PoW 协议[27]。

B.POW BASED ON [24]

B.基于[24]的功率

In Figure 6, we detail the Proof of Ownership (PoW) due to Halevi et al. [27].The protocol is divided into three phases: in the first phase, the file f is reduced into a buffer with maximum 64 MB in size.Then, the contents of the buffer are pseudo-randomly mixed and finally, the Merkle tree of the buffer is computed.The first two phases can be seen as applying a linear code to the file, and outputting a buffer Buf(f).Subsequently, to verify a PoW, the ver-ifier asks for the sibling paths of a random number of leaves from Buf(f) and checks that the authentication path matches the root of the Merkle tree.

在图 6 中，我们详细说明了哈莱维等人的所有权证明(PoW)[27]。该协议分为三个阶段:在第一阶段，文件 f 被缩减为最大 64 MB 大小的缓冲区。然后，对缓冲区的内容进行伪随机混合，最后计算缓冲区的 Merkle 树。前两个阶段可以看作是对文件应用一个线性代码，并输出一个缓冲区 Buf(f)。随后，为了验证一个 PoW，验证器从 Buf(f)中请求随机数量的叶子的兄弟路径，并检查验证路径是否与 Merkle 树的根匹配。

The buffer is an encoding of the file with a random linear code.The code is generated using SHA-256 to generate an array of pointer ptr .Due to the collision resistance of the hash function, any modi-fication of the file will therefore lead to a different code.Under the assumption that the code has a minimum distance of L/3, where L is the number of blocks in the buffer, any two valid buffers will be different in at least L/3 positions.Thus, the prover will only suc-ceed with a probability of (2/3)t in answering t challenges without knowing the correct buffer.For example, when t = 20 challenges are made, the probability of success is at most 2−.Note that with-out knowing the file in its entirety, two different random codes are derived in the first two steps due to the collision-resistance of the deployed hash function.In particular, it is highly unlikely that any block in the buffer can be predicted.We refer the readers to [27] for more details on the security of this construct.

缓冲区是用随机线性代码对文件进行编码。代码是使用 SHA-256 生成指针数组 ptr 生成的。由于散列函数的抗冲突性，文件的任何修改都会导致不同的代码。假设代码的最小距离为 L/3，其中 L 是缓冲区中的块数，任何两个有效缓冲区在至少 L/3 个位置上会有所不同。因此，在不知道正确的缓冲区的情况下，证明者在回答测试问题时只有(2/3)t 的概率。例如，当进行 t = 20 次挑战时，成功的概率最多为 2。注意，由于不知道文件的全部内容，由于部署的散列函数的抗冲突性，在前两个步骤中导出了两个不同的随机代码。特别是，缓冲区中的任何块都是不太可能被预测的。我们请读者参考[27]以获得关于该结构安全性的更多细节。

900