A Survey of Secure Data Deduplication Schemes for Cloud Storage Systems

YOUNGJOO SHIN, National Security Research Institute
DONGYOUNG KOO and JUNBEOM HUR, Korea University

Data deduplication has attracted many cloud service providers (CSPs) as a way to reduce storage costs. Even though the general deduplication approach has been increasingly accepted, it comes with many security and privacy problems due to the outsourced data delivery models of cloud storage. To deal with specific security and privacy issues, secure deduplication techniques have been proposed for cloud data, leading to a diverse range of solutions and trade-offs. Hence, in this article, we discuss ongoing research on secure deduplication for cloud data in consideration of the attack scenarios exploited most widely in cloud storage. On the basis of classification of deduplication system, we explore security risks and attack scenarios from both inside and outside adversaries. We then describe state-of-the-art secure deduplication techniques for each approach that deal with different security issues under specific or combined threat models, which include both cryptographic and protocol solutions. We discuss and compare each scheme in terms of security and efficiency specific to different security goals. Finally, we identify and discuss unresolved issues and further research challenges for secure deduplication in cloud storage.

CCS Concepts: • **Information systems → Deduplication**; **Cloud based storage**; • **Security and privacy → Public key encryption**; **Management and querying of encrypted data**; • **Networks → Cloud computing;**

Additional Key Words and Phrases: Message-dependent encryption, proof of ownership, traffic obfuscation, deterministic information dispersal

ACM Reference Format:
Youngjoo Shin, Dongyoung Koo, and Junbeom Hur. 2017. A survey of secure data deduplication schemes for cloud storage systems. ACM Comput. Surv. 49, 4, Article 74 (January 2017), 38 pages.
DOI: http://dx.doi.org/10.1145/3017428

1. INTRODUCTION

The cloud computing paradigm has significantly changed the management of data in small and medium businesses and personal data due to increased reliability, scalability, ubiquitous network access, rapid provisioning, and on-demand security control, all at negligible cost.

The CISCO Global Cloud Index (2012–2017) [Cisco 2015] showed that global data center traffic is expected to grow threefold and reach a total of 7.7 zettabytes annually by 2017. By 2019, global data center traffic will reach 10.4 zettabytes per year, 83% of all data center traffic will come from the cloud, and four out of five data center workloads will be processed in the cloud. With the unprecedented growth of cloud big

This work was supported by a Korea University Grant and by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (no. 2016R1A2A2A05005402).
Authors' addresses: Y. Shin, National Security Research Institute, 1559 Yuseongdae-ro, Yuseong-gu 34044, South Korea; email: yjshin@nsr.re.kr; D. Koo and J. Hur (corresponding authors), Department of Computer Science and Engineering, Korea University, 145 Anam-ro, South Korea; emails: {dykoo, jbhur}@korea.ac.kr.

DOI: http://dx.doi.org/10.1145/3017428

data, multiple copies of the same data exist within cloud storage as a massively increasing number of users independently store the same documents, media, or software packages.

Deduplication refers to a technique for eliminating redundant files or more fine-grain blocks of data. Deduplication identifies common data blocks or files and only stores a single instance. By doing so, deduplication can drastically reduce the cloud space needed to store large datasets. Recent studies have shown that interuser (or cross-user) data deduplication can reduce storage costs by more than 50% in standard file systems and by up to 90% to 95% for backup applications [Meyer and Bolosky 2012].

However, as data security is becoming one of the most important requirements for cloud computing services, the need for secure deduplication has been greatly increased for specific security goals in diverse cloud application scenarios. However, conventional cryptographic approaches, including encryption and deduplication, are opposed to one another to a great extent. Deduplication takes advantage of data similarity, whereas the goal of cryptography is semantic security, making ciphertext indistinguishable from random data. Preserving data security while benefiting from data deduplication remains a pivotal and challenging problem to be solved in cloud storage systems. Thus, the aim of secure deduplication is to provide both space efficiency and data security against both inside and outside adversaries.

To this end, secure deduplication techniques have been closely studied for specific security and efficiency goals in the presence of inside or outside adversaries regarding a diverse range of design decisions in terms of data granularity, deduplication location, duplicate check boundary, and system architecture. According to these four criteria corresponding to key design decisions, existing secure deduplication schemes can be categorized into cryptographic or security protocol–based approaches including message-dependent encryption, proof of ownership (PoW), traffic obfuscation, and deterministic information dispersal on the basis of their security goals.

Message-dependent encryption is mainly concerned about the confidentiality of the outsourced data in cloud storage. To achieve this security goal, it enables independent data owners to generate an encryption key from outsourced data. It allows the cloud storage to identify duplicate of ciphertexts that are encrypted by different users. *PoW* is a challenge-response protocol executed between a data uploader (or a user) and a cloud storage server. It allows a cloud server to verify whether a user who wants to outsource data really owns the data without requiring the data to be uploaded. *Traffic obfuscation* addresses security issues in client-side deduplication. In the presence of adversaries who can observe the pattern of traffic over the network during deduplication, traffic obfuscation–based solutions weaken their ability by obfuscating the pattern of network traffic, which makes it infeasible for them to learn any information from the observed network traffic. *Deterministic information dispersal* approaches address the confidentiality and availability of outsourced data in server-side deduplication systems in multicloud architecture. Dispersing data across multiple cloud storage servers improves the reliability of the outsourced data by adapting secret sharing algorithms as well as information dispersal algorithms (IDAs).

1.1. Scope and Contribution

This survey presents an extensive overview of secure deduplication systems and explains their design decisions and main challenges. This article is focused on secure deduplication in cloud storage—that is, we do not address general deduplication techniques without any security design or network deduplication, although some secure deduplication systems to which we refer do offer storage deduplication for plain data or network deduplication as a basic technique.

The first contribution of this survey is identifying four key design decisions common to all secure deduplication systems: data granularity, deduplication location, storage architecture, and the cloud service model. For each of these, we extend the existing taxonomy, define security threats from inside and outside adversaries, and describe the distinct approaches taken to address the main challenges of secure deduplication. As a second contribution, we categorize existing state-of-the-art solutions for secure deduplication into four distinct approaches: message-dependent encryption, PoW, traffic obfuscation, and deterministic information dispersal on the basis of their security goals. We then extensively analyze these with regard to security and efficiency, and discuss their advantages and disadvantages for each purpose from both theoretical and practical points of view. Finally, we also give insight into the challenges with regard to unresolved and open security issues as well as practical issues of secure deduplication schemes.

To the best of our knowledge, this is the first survey and discussion of secure deduplication techniques in cloud storage.

1.2. Related Work

There have been many extensive surveys of general data deduplication techniques. Mandagere et al. [2008] first explored the effectiveness and the efficiency trade-off among various deduplication systems in different conditions. In their work, by characterizing the taxonomy of available deduplication techniques, experimental evaluations were conducted using a real-world backup dataset.

On the basis of the taxonomy built by Mandagere et al. [2008], Paulo and Pereria [2014] further classified deduplication systems according to six key design decisions: data granularity, locality, timing, indexing, algorithm, and scope. Then, in the combination of design decisions, each deduplication technique has been analyzed with regard to the performance and the effectiveness in different storage environments, such as archival/backup storage and primary storage (e.g., HDD, SSD, and RAM).

By extension of Mandagere et al. [2008] and Paulo and Pereira [2014], Fu et al. [2015] presented more fine-grain taxonomy of deduplication techniques in terms of the following functional and operational aspects: key value, fingerprint prefetching, segmenting, sampling, rewriting, and restore. They also proposed a general-purpose framework for evaluating deduplication and exploring trade-offs among backup performance and storage costs.

As another work, Meyer and Bolosky [2012] studied the practical impact of data deduplication to the system using nearly 1,000 Windows file systems. They compared efficiency in terms of the storage cost between whole-file deduplication and block-level deduplication.

Most of the previous survey works have focused on efficiency and effectiveness analysis of data deduplication schemes without any security considerations. Unlike previous surveys, this article mainly focuses on identifying various security threats with regard to data confidentiality, integrity, and availability, and notable attacks on the deduplication system in cloud storage. The article also addresses classifying the existing secure deduplication techniques and analyzing them to mitigate these attacks. The contribution of this work can be found in its originality that differs from these previous survey works, which focus on exploring effectiveness and performance trade-offs of general deduplication techniques without any security considerations.

1.3. Organization

The remainder of the article is organized as follows. Section 2 provides a deduplication system overview, including the aspects of data granularity, location, cloud system architecture, and the cloud service model. Section 3 describes security threat models

to cloud storage by both inside and outside adversaries. Section 4 defines the criteria for the measurement of the security and efficiency of secure deduplication techniques. Section 5 surveys state-of-the-art solutions for secure deduplication, which are classified into message-dependent encryption, PoW, traffic obfuscation, and deterministic information dispersal. It then presents the analysis and comparison results for each secure deduplication scheme with regard to security and efficiency. Sections 6 and 7 discuss the current research focus on unresolved security issues for secure deduplication and its practical limitations, respectively. Section 8 presents final remarks about the survey.

2. DEDUPLICATION DESIGN CRITERIA

In a cloud storage system, data deduplication is usually performed by entities: a client (or a user), who is the data owner, and a cloud service provider (CSP), which provides storage space for the outsourced data. Data deduplication schemes for cloud storage may differ according to security goals and design criteria. In this section, we classify these schemes based on the criteria built by Mandagere et al. [2008] and Paulo and Pereira [2014]: data granularity, deduplication location, duplicate check boundaries, and system architecture.

2.1. Data Granularity

There are various ways to partition data into basic units for the elimination of duplicates. In file-level deduplication, which is one of the most straightforward approaches, a file is treated as the basic unit of deduplication. Hash signatures are computed from the content of the entire file and then used to find duplicate files in the cloud by comparing their hash signatures. Block-level deduplication is another common approach, in which a file is divided into multiple blocks and duplicates are checked for each block. There are two different approaches to dividing files into blocks: fixed-size chunking and variable-size chunking. In fixed-size chunking, every file is divided into fixed-size blocks, and the hash algorithm used in file-based deduplication is applied to each block. On the other hand, variable-size chunking uses Rabin fingerprinting [Broder 2000; Rabin 1981] as an algorithm to generate hash signatures. The Rabin fingerprinting algorithm considers a file as a stream of bytes and looks for prespecified delimiters in the byte stream. Each data chunk separated by the delimiters is then treated as the basic unit of deduplication.

Block-level deduplication usually has a higher deduplication ratio than file-level deduplication but incurs considerable overheads due to the large amount of metadata generated for each block. In contrast, file-level deduplication offers relatively low overhead in maintaining the metadata [Meyer and Bolosky 2012; Harnik et al. 2012].

2.2. Deduplication Location

Data deduplication can be performed on the server side, client side, and gateway side. In server-side deduplication, clients who want to outsource their data would always upload it to a cloud storage server over the network. Once the CSP receives the data, it performs deduplication by finding duplicates within the stored data and eliminating them.

In the client-side deduplication approach, a client first computes a hash value for the data to be uploaded and sends it to the CSP before uploading the actual content of the data. The CSP then checks whether the same hash value exists in storage. If an identical hash is found, then the server cancels the actual upload and marks the client as an owner of the existing data. Otherwise, the client continues to upload the data to the server. Compared to server-side deduplication, this approach has an advantage in terms of reducing network bandwidth consumption [Bobbarjung et al. 2006].

In some cloud computing environments, particularly in hybrid cloud computing models, a certain type of appliance server, referred to as a storage gateway, is usually deployed on a customer's premise private network. A storage gateway provides a customer with access to a public cloud storage service. Gateway-side deduplication utilizes storage gateways to execute deduplication. On receiving outsourcing data from a client, the storage gateway performs deduplication along with a CSP on behalf of the client.

2.3. Duplicate Check Boundary

Based on the duplicate check boundary, deduplication techniques can be categorized into intrauser and interuser approaches. When finding duplicates in uploaded data, intrauser deduplication solely considers data in the cloud storage that has been outsourced by the same data owner. On the other hand, interuser deduplication considers all stored data across multiple data owners in the cloud storage.

Both intrauser and interuser deduplication techniques effectively eliminate redundant data in cloud storage. Since intrauser deduplication finds and removes duplicate copies of the same users' data, this technique is more useful for backup systems, where duplicates are likely to be found among repeated backups made by a single user [Kaczmarczyk et al. 2012]. Interuser deduplication is effective for storage systems within an organization where a large number of duplicates exists among data owned by different users, such as business files, VM images [Jin and Miller 2009], and snapshots of workstation file systems [Meyer and Bolosky 2012].

2.4. System Architecture

Secure deduplication schemes can be built on two different cloud system architectures: (1) single cloud architecture, in which a single CSP constructs and provides a storage service, and (2) multicloud architecture, also referred to as cloud-of-clouds architecture, in which a cloud storage service is built on multiple CSPs and users' data disperses across the multiple cloud storages.

Single cloud architecture is the prevalent approach for many commercial CSPs (e.g., Dropbox, Google Drive, and Mozy); however, this suffers from potential problems, such as data loss due to system failure and vendor lock-in [Abu-Libdeh et al. 2010]. Multicloud architecture tries to avoid a single point of failure in that data is replicated [Abu-Libdeh et al. 2010] and divided into multiple shares [Li et al. 2015b; Ling and Datta 2014], and then the shares are dispersed across multiple independent cloud storages. Data deduplication techniques for multicloud architecture such as RAIDer [Ling and Datta 2014] and CDStore [Li et al. 2015b] handle these distributed shares, which are usually processed by erasure codes like the Reed-Solomon code, as a basic unit for checking duplicates.

3. SECURITY RISKS TO CLOUD STORAGE WITH DEDUPLICATION

A CSP owns the cloud infrastructure, including storage disks and the network for offering a data outsourcing service, whereas outsourced data is held by data owners. This separation of ownership between the data and the infrastructure causes data owners (i.e., clients) to lose physical control over their data after outsourcing it and thus generates security concerns regarding the security and privacy of the outsourced data. In particular, a cloud storage system that employs deduplication techniques faces more severe and various security risks, as the deduplication itself becomes a means by which adversaries can mount their attacks. In the general threat model for cloud computing, two types of adversary are considered:

—*Inside adversaries*: These are the adversaries within a CSP. They could be malicious employees of a CSP with administrative privileges for a storage system, network,

or the other infrastructure, and even the cloud service provider itself. A hacker from outside who compromises and takes control of the cloud storage system is also considered an inside adversary. These inside adversaries have an interest in sensitive data stored in the cloud storage, the access to which is supposed to be prohibited for everyone except a data owner. To learn information in the data, they can launch sophisticated attacks, including utilization of deduplication as leverage. In addition, inside adversaries might also be interested in breaking the integrity of the data in the storage for the purpose of denial of service (DoS) or other goals.

—*Outside adversaries*: These are the adversaries outside a CSP. They could be malicious users who are not authorized to access outsourced data owned by other users but are interested in obtaining sensitive information in the data. We also consider any hackers who can attack legitimate users and masquerade as them to launch attacks as outside adversaries. As with inside adversaries, a primary goal of outside adversaries is to obtain useful information from the outsourced data in the cloud storage. To achieve this attack, they may exploit data deduplication protocols. They are also interested in DoS by corrupting data as well as abusing the cloud storage service for other malicious purposes.

In the following section, we introduce various security risks that are caused by inside and outside adversaries in cloud storage systems utilizing deduplication techniques.

3.1. Threats from Insider Attacks

Outsourced data in cloud storage is exposed to threats from inside adversaries, who will take advantage of their ability to control cloud service platforms and backend infrastructure to access data.

3.1.1. Data Breaches. CSPs are usually reluctant to apply encryption algorithms with user-supplied keys to the stored data, as it makes deduplication impossible even for the same data. In other words, using distinct keys, a conventional encryption algorithm (e.g., AES) will yield totally different ciphertexts for identical messages, which makes if difficult to check whether these plaintexts are the same based on the ciphertexts. Thus, to enable deduplication and reduce cloud storage costs, outsourced data is handled without applying encryption. This can eventually lead to data breaches by curious inside adversaries who can access the plain data in the cloud storage. To address the conflict issues regarding confidentiality and storage efficiency, most cloud storage services, such as Dropbox or AWS S3, encrypt the data under encryption keys selected by themselves, not by data owners. This approach enables data deduplication of the encrypted data. In our threat model, however, it is still considered to be vulnerable to insider attacks since malicious CSPs are able to access the plain data in the cloud storage and obtain private information from it.

3.1.2. Data Loss. Once data has been uploaded to cloud storage, data owners lose control over their data. The reliability of the outsourced data depends on how tolerant the cloud infrastructure is to system faults. However, despite huge efforts by CSPs to offer seamless and faultless service, it is inevitable that cloud service outages will occasionally occur (e.g., Dropbox suffered an outage for 10 hours in 2013). Such outages may inflict severe damage on the storage system and eventually make the data affected by the damage unrecoverable. Data deduplication further increases concerns about data reliability due to the nature of deduplication in which only one copy of the data is kept regardless of the number of uploads.

In addition to the failure of the storage system, data loss can be intentionally caused by inside adversaries, especially by a malicious CSP. In other words, to keep disk space sufficient without increasing any cost, they may illegally erase a portion of data in the

storage that has not been recently accessed by clients without notifying the clients of the deletion. Other inside adversaries, such as hackers, who aim to cause DoS for the cloud storage service may also damage the data in the cloud storage.

3.2. Threats from Outsider Attacks

Outside adversaries are able to exploit data deduplication as leverage to launch attacks on cloud storage services. In this section, we present various security threats caused by outside adversaries on cloud storage with deduplication.

3.2.1. Information Leakage Through Side Channels. As shown in recent studies [Harnik et al. 2010; Mulazzani et al. 2011; Pulls 2012], client-side cross-user data deduplication protocols can be exploited as a side channel through which outside adversaries such as a malicious user (or client) can learn information from other users' sensitive data. A side channel is established via two inherent properties of client-side deduplication: (1) data transmission over a network is visible to an adversary, and (2) fingerprints for uploading data (e.g., a hash for the data) can be used to determine the existence of copies on the cloud server. By exploiting side channels, the adversary can plausibly mount the following attacks:

—*Identifying the existence of specific data*: Suppose that an adversary wants to confirm the existence of interesting data, say file F, in the storage. The adversary can easily confirm the existence of F by uploading F and monitoring its network activity. If the entire content of F is uploaded through the network, then the adversary will learn that F does not exist in the cloud storage. Otherwise, it can be assumed that F has already been uploaded to the storage.
—*Learning the content of the data*: An adversary who is eager to learn the content of interesting file F can mount an online-guessing attack by utilizing the side channel. Using a dictionary that contains multiple candidates for F, the adversary repeatedly runs a data deduplication protocol with a CSP for each guessed F' of F until he or she observes a deduplication event.

3.2.2. Unauthorized Data Access. In addition to information leakage by means of side channels, client-side data deduplication is also vulnerable to another kind of attack that allows unauthorized outside adversaries to access other users' sensitive data in the cloud storage [Halevi et al. 2011]. Prior to uploading data to cloud storage, client-side deduplication systems require a user to send a small fingerprint of the data to the cloud server, which is typically a hash value for the data. Using this fingerprint as a search key, the CSP tries to find stored data with the same fingerprint. If matching data is found, then the upload of the data is skipped and the CSP grants ownership of the data to the uploader (user). A new security threat originates from the fact that a CSP verifies ownership of the outsourced data by using a hash value that is usually not considered a secret, especially when the data is predictable. Hence, even without the actual data, malicious users are able to prove ownership to the CSP with a guessed hash value. An outside adversary may exploit this vulnerability to take ownership of interesting data that they are not supposed to be authorized to access. By presenting its hash, the adversary can convince the CSP that he or she actually owns the data and thus will be able to download it from the server.

Mulazzani et al. [2011] discovered the aforementioned vulnerability on Dropbox, which is one of the largest commercial CSPs. To demonstrate this attack in Dropbox, they modified a crypto module of a Dropbox client program so that instead of normal hash computation, a specific hash value for an interesting file is given to the Dropbox server, to which the client is not authorized to access. By running this modified client

program, they have found that Dropbox does not detect the manipulation of hash values and even enables arbitrary access to any files on the storage by the client.

3.2.3. Abuse of Cloud Storage Services. By means of the aforementioned security vulnerabilities in client-side deduplication, outside adversaries are even able to abuse cloud storage services for other malicious purposes. The following scenarios demonstrate how they can abuse a cloud storage service:

—*Establishing a covert channel*: Suppose that two adversaries $\mathcal{A}$ and $\mathcal{B}$ seek to communicate with each other in secrecy using a side channel incurred by (client-side) deduplication. They agree to use two distinct random files F_0 and F_1 to transfer one-bit data to the other party. To send '1,' an adversary, say $\mathcal{A}$, would upload F_1 to cloud storage. After a certain amount of time, $\mathcal{B}$ uploads F_1 and observes its deduplication event over the network. If the deduplication of F_1 occurs, then $\mathcal{B}$ eventually learns that '1' was sent from $\mathcal{A}$. In the same way, $\mathcal{A}$ is also able to send '0' to $\mathcal{B}$ using the other file F_0.
—*Use as a content delivery network*: A cloud storage service that utilizes client-side deduplication can be abused as a content delivery network (CDN). Suppose that adversary $\mathcal{A}$ has a large file F, which probably contains copyright-violating content (e.g., a pirated movie), and he or she wishes to transfer F to another adversary $\mathcal{B}$. Instead of using a legitimate CDN to deliver F, A would upload F to a CSP. Once the upload of F has been completed, A sends the corresponding hash value of F to B via an arbitrary channel. B can then easily obtain ownership of F by just presenting the hash of F to the CSP and download the whole content of F. As noted in Halevi et al. [2011], this behavior of $\mathcal{A}$ and $\mathcal{B}$ conflicts with the business model of cloud storage services, which is meant to mainly support large uploads (e.g., data backups) rather than downloads.

3.2.4. Data Poisoning. In the general deduplication process, an initial uploader (i.e., a client who uploads data for the first time), should send both the data and the corresponding tag to the CSP. By using the tag as a search key, the CSP is able to check for duplicates of this uploaded data. However, if the data is outsourced in an encrypted form, then the CSP cannot verify the consistency between the encrypted data and the corresponding tag. This problem makes the deduplication system susceptible to a data poisoning attack. For example, suppose that an outside adversary changes original data file F to F' and then uploads F' and a tag that corresponds with F. Then, the same data file F outsourced by subsequent legitimate data owners will be deduplicated by the CSP. Afterward, whenever they want to download F from the CSP, they will get the maliciously modified data file F' rather than the original F.

4. TAXONOMY OF SECURE DEDUPLICATION SCHEMES AND EVALUATION CRITERIA

A secure deduplication scheme has the goal of mitigating the aforementioned security threats and thus provides a solution that allows cloud storage to securely take advantage of deduplication techniques. The diversity of deduplication systems, as classified in Section 2, and the many possible attacks on cloud storage from inside/outside adversaries have led to various secure deduplication solutions in the literature. Based on key ideas and design decisions to achieve security goals, we categorize state-of-the-art secure deduplication schemes into four different approaches: message-dependent encryption, PoW, traffic obfuscation, and deterministic information dispersal. Each approach covers different areas of the deduplication systems, and thus the security threats that they address vary accordingly. Details of secure deduplication approaches will be presented in Section 5. In this section, we briefly present our taxonomy of secure deduplication schemes and their evaluation criteria.

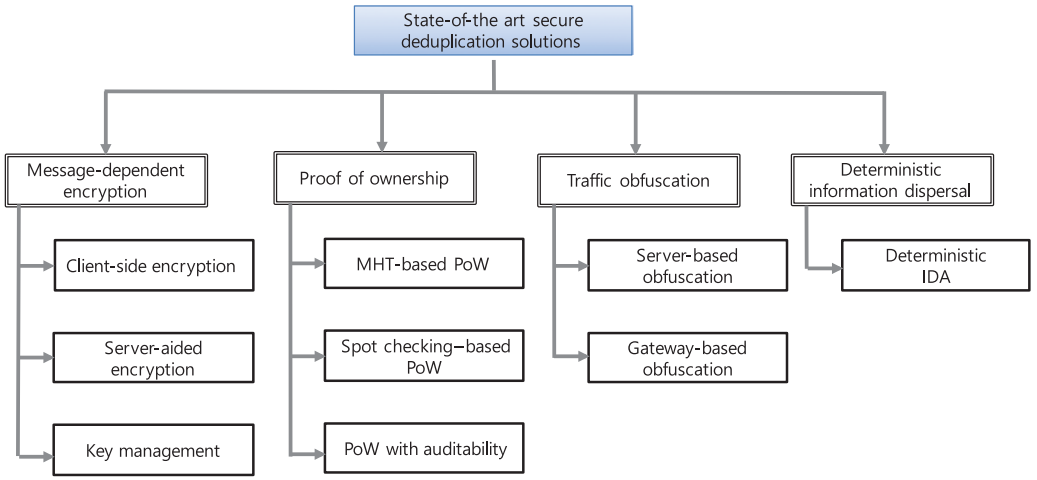

Fig. 1. Taxonomy of state-of-the-art secure deduplication solutions.

4.1. Classification

Figure 1 illustrates a taxonomy of state-of-the-art secure deduplication schemes. The classification is constructed according to the key idea that each method adopts to address the security issues of deduplication systems:

—*Message-dependent encryption*: This approach mainly concerns protecting the confidentiality of outsourced data against untrusted and malicious CSPs. To achieve this security goal, encryption techniques are utilized in constructing a scheme. However, instead of following the randomized key generation of conventional encryption algorithms, this approach deterministically computes an encryption key from an information source that depends on the outsourcing data so as to allow a CSP to find duplicate ciphertexts encrypted by multiple users. According to the main design decision regarding message-dependent key generation, schemes in this approach can be classified into three subcategories: client-side encryption, server-aided encryption, and key management. The systems that are covered by most message-dependent encryption schemes are largely confined to deduplication systems based on the single cloud architecture.

—*Proof of ownership*: This approach focuses on addressing various security issues in client-side deduplication systems. PoW is basically a challenge-response protocol that is executed between an outsourcing user and a CSP. PoW allows a CSP to verify whether a user who wants to outsource data really owns it without having to upload the entire content of the data. Deduplication systems equipped with PoW are supposed to be secure against insider/outsider attacks including data breaches, data loss, unauthorized access, and abuse of service. With respect to the data structure used in the protocol and the features that they provide, PoW-based solutions can be classified into three subcategories: Merkle hash tree (MHT)-based PoW, spot checking–based PoW, and PoW with auditability. PoW-based schemes usually consider providing security in client-side deduplication systems that support duplicate checks across multiple users (i.e., an interuser duplicate check boundary).

—*Traffic obfuscation*: This approach addresses security issues in client-side deduplication, specifically threats including the abuse of cloud storage and other attacks exploiting side channels. Security threats from outside adversaries originate from the nature of client-side deduplication that allows them to observe patterns of network

traffic during the execution of a deduplication protocol. Traffic obfuscation–based solutions try to weaken this ability by obfuscating these patterns in such a way that it is infeasible to learn any information by observing network traffic. Based on the location where the obfuscating algorithm is run, this scheme can be classified into server-based obfuscation and gateway-based obfuscation. The traffic obfuscation approach usually covers client-side deduplication systems with an interuser duplicate check boundary.

—*Deterministic information dispersal*: This approach addresses both the preservation of confidentiality and the availability of outsourced data in cloud storage, specifically for server-side deduplication systems on a multicloud architecture. In cloud storage services over multiple CSPs, the availability of outsourced data can be improved by dispersing the data across multiple cloud storage servers. However, data dispersion makes the deduplication of encrypted data a challenging problem. Deterministic information dispersal solves this problem by applying secret sharing techniques instead of encryption algorithms. Similar to message-dependent encryption, this approach replaces randomized elements with a value that is deterministically computed from the data so as to allow each CSP to find duplicate copies of dispersed data.

4.2. Evaluation Criteria

Secure deduplication schemes are evaluated in terms of two factors: (a) the security properties that each scheme provides and (b) the system overheads incurred on a CSP and a client.

Regarding the security threats presented in Section 3, a secure deduplication scheme is required to satisfy the following security properties:

(1) *Confidentiality*: The original data content outsourced to cloud storage should not be revealed to anyone except the user who owns the data.
(2) *Availability*: The outsourced data should always be available to users who own the data, even during system failure.
(3) *Side-channel resistance*: In client-side deduplication systems, it should be impossible for adversaries to learn any information about outsourced data by observing deduplication traffic over the network.
(4) *Ownership authenticity*: In client-side deduplication systems, only a valid client (or a user) who possesses the original data should be verified as an authentic owner of the outsourced data in the cloud storage.
(5) *Data integrity*: A client or CSP should be able to verify that the outsourced data has not been modified or deleted by adversaries.

Applying secure deduplication may lead to some unavoidable costs for CSPs. Hence, each scheme is also evaluated with respect to performance and efficiency in the following aspects: (1) computation overhead, (2) storage overhead, and (3) communication overhead.

5. STATE-OF-THE-ART SOLUTIONS FOR SECURE DEDUPLICATION

According to the identified requirements in terms of security properties and system overhead in the previous section, we analyze state-of-the-art secure deduplication solutions, which can be categorized into four approaches: message-dependent encryption, PoW, traffic obfuscation, and deterministic information dispersal. We give details on the secure deduplication solutions, present representative schemes for each approach, and discuss the advantages and disadvantages of these solutions through comparisons.

5.1. Message-Dependent Encryption

In cloud storage systems, data owners (or users) lose control of their data after outsourcing it to cloud storage. In practical scenarios, however, the CSP cannot be fully trusted (e.g., it can leak the clients' private data for monetary purposes or manipulate it to ease storage costs). Therefore, data owners may encrypt their data before outsourcing for the sake of data confidentiality. Unfortunately, conventional data encryption makes deduplication very difficult for the following two reasons: (1) the CSP finds it difficult to identify whether the underlying plain data is the same or not, as data encryption under different keys selected by different users generates different ciphertexts, and (2) even though the CSP can determine the identity of the underlying data, it is difficult to store two different ciphertexts in a storage-efficient manner so that any data owner can retrieve the plain data after outsourcing it.

In this respect, message-dependent encryption is considered a promising cryptographic primitive for ensuring data privacy during deduplication. In message-dependent encryption, a data owner computes an encryption key from the data about to be encrypted and encrypts the data under the key. Formally, it can be defined as the following four deterministic algorithms (key generation, encryption, decryption, and tag generation):

—$ck_F \leftarrow KeyGen(F)$: The key generation algorithm takes a data content F as input and outputs the symmetric key ck_F of F. Basically, the key ck_F is generated by rendering a cryptographic hash function h such that $ck_F = h(F)$.
—$ct_F \leftarrow Encrypt(ck_F, F)$: The encryption algorithm takes the key ck_F and the data content F as input and outputs a ciphertext ct_F encrypted under ck_F.
—$F \leftarrow Decrypt(ck_F, ct_F)$: The decryption algorithm takes the key ck_F and the encrypted content ct_F and outputs the restored plain data F.
—$tag_F \leftarrow TagGen(F)$: The tag generation algorithm takes the data content F as input and outputs a tag tag_F of F.

All of the generated values from the preceding four algorithms are the same for the same F in generic message–dependent encryption. Therefore, multiple data owners of the same data content are allowed to generate the same keys (and eventually the same ciphertext) without additional communication for key agreement before outsourcing.

With the generic message–dependent encryption described earlier as a baseline, there have been numerous studies looking at the enhancement of security and functionality. In implementing the *KeyGen* algorithm, message-dependent encryption can be further categorized into three groups: client-side encryption, server-aided encryption, and key management. In secure deduplication using client-side encryption, an encryption key is generated solely by the data owner before outsourcing. However, it is widely known that a key made by a single entity is susceptible to offline brute-force attacks on predictable message space, so an adversary can try all possible combinations to learn the underlying plain data. To mitigate offline and online brute-force dictionary attacks, server-aided encryption has been introduced. In this, the key is generated through secure multiparty computation with an independent key server (KS) rather than by a single entity. Key management approaches supporting secure deduplication have also been studied to securely and efficiently manage the generated key produced by the *KeyGen* algorithm.

5.1.1. Client-Side Encryption. Bellare et al. [2013a] first introduced the generalized framework of generic client-side message-dependent encryption (as message-locked encryption (MLE)) and demonstrated its practicality and theoretical soundness. MLE syntax is identical to generic message–dependent encryption algorithms such as key generation K, encryption E, decryption D, and tag generation T. When it comes to

$K = KeyGen(F)$, $E = Encrypt(ck_F, F)$, $D = Decrypt(ck_F, ct_F)$, and $T = TagGen(F)$, it can be viewed as convergent encryption (CE) [Douceur et al. 2002]. For practicality, variants of message-dependent encryptions can be categorized into four groups (CE, HCE1, HCE2, and RCE) and analyzed in terms of relative efficiency and tag consistency. Tag consistency (TC) means that no computationally bounded adversary can create a valid tag for F, whereas the outsourced ciphertext is an encryption of F' such that $F' \neq F$, and strong tag consistency (STC) is TC where the outsourced ciphertext cannot be decrypted correctly. For secure symmetric key encryption SE, decryption SD, and concatenation $\|$, CE is composed of $K = h(F)$, $E = SE(K, F)$, $D = SD(K, SE(K, F))$, and $T = h(C)$ with STC, whereas HCE1 is composed of $K = h(F)$, $E = SE(K, F) \| h(K)$, $D = SD(SE(K, F))$, and $T = h(K)$ without TC. HCE2 and RCE are newly introduced groups that support TC (but not STC), where the efficiency of HCE2 is identical to that of HCE1, whereas RCE is the most efficient due to the use of a random salt in key generation. From a theoretical aspect, the security of MLE and its relationship with other cryptographic primitives are provided. Unfortunately, MLE-style deduplication schemes cannot guarantee semantic security because message-dependent encryption derives an encryption key from the message itself. In other words, an adversary can find the underlying plain data by mounting a brute-force attack on the entire message space. In this context, MLE provides semantic security for unpredictable messages with high min-entropy. By showing that an encryption of a message is indistinguishable from random elements in ciphertext space, MLE provides privacy against chosen-distribution attack (PRV-CDA). With its relationship with deterministic public key encryption and correlated-input hash functions, the authors demonstrated that MLE can be constructed from these primitives.

Abadi et al. [2013] further strengthened Bellare et al.'s security definitions by considering plaintext distributions that might be dependent on system security parameters. They pointed out that an implicit constraint in which the distribution of plaintexts can be chosen by an adversary can jeopardize the security of MLE in the sense that the independence of plaintext distribution cannot be guaranteed. To improve the security of parameter-dependent messages, termed *lock-dependent security*, they provided two alternative approaches that have a tag, ciphertext, and proof of consistency. First, instead of deterministic derivation of a tag for duplicate checking, a fully randomized scheme supporting duplicate checking of the outsourced ciphertext is introduced. Specifically, the tag for file F is computed as $\tau = (g^r, g^{r \cdot h_1(F)})$, and the duplicate check between two tags $\tau_1 = (\tau_{1,1}, \tau_{1,2})$ and $\tau_2 = (\tau_{2,1}, \tau_{2,2})$ is achieved by testing $e(\tau_{1,1}, \tau_{2,2}) \stackrel{?}{=} e(\tau_{2,1}, \tau_{1,2})$ for a uniform random $r \in \mathbb{Z}_p$, symmetric bilinear map $e : G \times G \rightarrow G_T$ defined over groups G and G_T with generator g of G, and collision-resistant hash function $h_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. For the security of the ciphertext and proof of consistency, a combination of a secret sharing scheme (SSS) and a cut-and-choose technique is employed, which requires a small additive storage overhead independent of message length. The second approach is to bound the computational power of the adversarial message distributions to a predetermined threshold while still using deterministic tags. While using semantically secure encryption $SE(k, F \| r)$, where k and r are derived deterministically from F, key k and randomness r are computed as $k = \bigoplus_{i=1}^{q+1} H_1(m \| i)$ and $r = \bigoplus_{i=1}^{q+1} H_2(m \| i)$, where H_1 and H_2 are two independent hash functions. The pseudorandomness of k and r provides security against q-bounded queries by an adversary in a random oracle model.

Bellare and Keelveedhi [2015] also extended MLE for secure data deduplication in an interactive manner, called *interactive MLE* (iMLE). This approach provides semantic security for messages that are correlated as well as dependent on the public system parameters in a standard model by exploiting fully homomorphic encryption. They

presented the theoretical possibility of providing semantic security for secure deduplication, although this cannot be adopted in real-world scenarios at present because it makes extensive use of fully homomorphic encryption, which requires extremely high computation overhead.

In MLE-based secure deduplication, data owners who want to upload data can learn whether the data has already been outsourced by another data owner or not. On the other hand, Liu et al. [2015] presented client-side encryption for secure deduplication without revealing the existence of outsourcing data in cloud storage. By exploiting password authenticated key exchange (PAKE) in an oblivious key sharing protocol between the uploading data owner and previously outsourced data owners, only a valid data owner can reconstruct the encryption key (i.e., the password in PAKE) for the outsourced ciphertext. PAKE can be viewed as an extension of the Diffie-Hellman key agreement protocol based on a CDH assumption in pairwise one-to-many participants. Specifically, when the short hash (permitting numerous collisions) of a file is sent to the CSP by the data owner, the CSP notifies data owners who previously uploaded the file with the same short hash value and relays messages exchanged in a single-round PAKE protocol run by the uploading and uploaded data owners. From the pairwise established session keys, the CSP performs deduplication in cases where a pair of session keys is the same. Otherwise, it stores the encryption independently without notification to the owner. This incurs additional computation overhead for previously outsourced data owners, which is inherited from a different assumption to the one for MLE in that previously outsourced data owners do not need to participate in key agreement as in the PAKE-based approach.

5.1.2. Server-Aided Encryption. For malicious inside adversaries of the CSP, the aforementioned MLE-based client-side encryptions are inherently susceptible to offline brute-force attacks on predictable message spaces. Owing to the deterministic key generation and keyless properties of MLE, such approaches are unable to guarantee semantic security against adversaries who attempt exhaustive searches on a known set. Although RCE randomizes the plaintext (and, accordingly, the ciphertext) in the encryption phase, it provides confidentiality only for unpredictable messages, clearly failing to achieve semantic security. To mitigate the lack of security for unpredictable message spaces, which are too large to exhaust, recent studies introduced additional servers involved in key generation for the deduplication of encrypted data. Bellare et al. [2013b] proposed server-aided deduplication for encrypted data, called *DupLESS*, which provides security against brute-force attacks by introducing an additional KS. DupLESS and MLE share the principle that the message-derived key and the resulting ciphertext, respectively, should coincide for the same content. However, instead of a deterministic hash of messages, DupLESS allows data owners to acquire the message-derived key through an oblivious PRF (OPRF) protocol with the KS. More specifically, a data owner first computes and sends a blinded hash of file F, $x = H(F) \cdot r^e \ mod \ N$, to the KS for hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_N$, RSA encryption key e, and RSA modulus N. Then, the KS possessing system-wide secret key $sk = (N, d)$ generates blinded signature $y = x^d \ mod \ N$ and returns it to the owner. The owner can derive message-derived key $G(z)$ for hash function $G : \mathbb{Z}_N \rightarrow \{0,1\}^k$ with the size of encryption key k only if blinding-removed signature $z = y \cdot r^e \ mod \ N$ is the same as $H(F)$. Using RSA-based OPRF in key generation, the KS learns nothing about the owner's input nor the output of the OPRF protocol, and the owner learns nothing about the KS's secret key, which eliminates a security vulnerability for the KS. Moreover, this involvement of the KS in key generation makes DupLESS more resistant to a brute-force attack since the KS plays the role of a rate-limiting control center. DupLESS provides deduplication

against online brute-force attacks, which leads to stronger guarantees of confidentiality than for the MLEs.

The security of DupLESS remains at the same level as that of MLE in cases where the CSP colludes with the KS (i.e., a compromise of the KS). To prevent a single point of failure for the KS, Miao et al. [2015] introduced multiserver-aided DupLESS based on threshold blind signatures. This scheme allows data owners to generate an encryption key from the interaction with multiple KSs. In this situation, multiple KSs generate their own secret share in the setup phase by exploiting the noninteractive verifiable distributed SSS proposed by Pedersen [1992]. In other words, each KS, say KS_i, randomly selects its secret share s_i and generates random polynomial F_i with degree $t-1$ such that $F_i(0) = s_i$. Then, KS_i distributes $g^{a_{i,j}}$ for every j-th coefficient $a_{i,j}$ $(0 \leq j < t)$ of F_i and securely delivers $F_i(l)$ to the l-th KS for $0 \leq l \leq n$. For system-wide secret key $S = \sum_{i=1}^{n} s_i$ and public key $P = g^s$, the secret/public key pair of each KS_i becomes $S_i = \sum_{l=0}^{n} F_l(i)$ and $P_i = g^{S_i}$, respectively. In key generation, a data owner requests by running a threshold blind signature protocol with multiple KSs. After sending $F' = r \cdot h(F)$ for random nonce $r \xleftarrow{\$} \mathbb{Z}_p^*$ with $W_i = \prod_{l \neq i \in C} \frac{l}{l-i}$ to KS_i where $C \subseteq \{1, \ldots, n\}$ and $|C| = t$ to KS_i, the data owner receives $\sigma_i = (F')^{S_i \cdot W_i}$ as a partial signature. By combining t partial blinded signatures, the owner can calculate the encryption key $K = (\prod_{i \in C} \sigma_i)^{r^{-1}}$ for file F. On receiving $h(SE(K, F))$ as the tag for duplicate checking, the CSP can determine whether to deduplicate or to store it independently. By adopting (n, t)-SSS among multiple KSs, the system avoids the single point of failure of the KS where partial KSs cannot learn the distributed secret key. Duan [2014] presented deduplication of encrypted data by introducing a distributed key generation protocol. To avoid the single point of failure of a single KS, a trusted dealer distributes key shares for other online users in the system. Based on Shoup's RSA noninteractive threshold signature technique, shares of a random secret signing key are distributed to n online signers by the dealer. Through the polynomial interpolation of (blinded) partial signatures, a data owner can produce a complete signature as the common key to be used in the encryption of file F. In this scheme, the trusted dealer plays a similar role to that of the KS in DupLESS, but the task of independent servers in Miao et al.'s work is leveraged to distributed online signers in such a way that a data owner interacts with multiple signers to acquire partial signatures. Multiple users are supposed to participate in key generation from the upload request for the encryption of the outsourced data, and thus this approach is more suited to peer-to-peer (P2P) environments than cloud storage services where multiple users do not have to always be online.

Li et al. [2015a] proposed an authorized deduplication system with differential privileges in a hybrid cloud architecture. The term *hybrid* in a scheme implies that a private cloud is in charge of the management of secret keys for data owners with differential privileges and the issuing of searching tokens according to these privileges, whereas a public cloud is responsible for duplicate checking and archival/retrieval of outsourced data. Specifically, a data owner with privilege p_τ acquires token $\phi = h(TagGen(F), k_{p_\tau})$ for the secret key k_{p_τ} of the privilege p_τ from the private cloud (without the actual transmission of k_{p_τ}) after authentication of the affiliated owner. By allowing only authorized data owners to send valid requests for files with the corresponding access policy, the public cloud just checks for duplicates in its storage with sufficient privileges. The owner is provided a pointer for the file if duplicates are found and undergoes the independent ownership proof process otherwise. This tag generation method also efficiently prevents brute-force attacks launched by the public cloud or dishonest users because it is based on two independent cloud architectures.

On the another hand, studies on context-aware secure deduplication with the aid of additional servers have been conducted for a variety of applications. Liu et al. [2014] proposed policy-based block-level deduplication making use of proxy re-encryption. When exploiting the key encapsulation technique, the deduplication module is separated from cloud storage to reduce the possibility of both the key and ciphertext being compromised at the same time. In this approach, data owners can determine the type of outsourcing data based on security criteria: not encrypted, encrypted with deduplication, or encrypted without deduplication. For encrypted data allowing deduplication, the data owners are supposed to upload their encrypted data to a deduplication proxy, where the duplicate data is re-encrypted with the aid of the KS. Specifically, given key pair $pk_A = (Z^{a_1}, g^{a_2})$ and $sk_A = (a_1, a_2)$ of user $\mathcal{A}$ (with bilinear map $e : G \times G \rightarrow G_T$, generator g of G, $Z = e(g, g)$, and uniform random values $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_p^*$), $\mathcal{A}$ generates three ciphertexts for each block B_i constituting file F, such as $C_{A,1} = (Z^{a_1 \cdot k_i}, m \cdot Z^{k_i})$, $C_{A,2} = (Z^{a_2 \cdot k_i}, m \cdot Z^{k_i})$, and $C_{A,r} = (g^{k_i}, m \cdot Z^{a_1 \cdot k_i})$ for uniform randomly selected key $k_i \xleftarrow{\$} \mathbb{Z}_p$. Upon receipt of $\langle h(B_i), C_{A,1}, C_{A,2}, C_{A,r} \rangle$ from $\mathcal{A}$, the proxy checks duplication based on $h(B_i)$ and performs re-encryption for the duplicates already outsourced by $\mathcal{B}$ using re-encryption key $rk_{B \rightarrow A} = g^{b_1 \cdot a_2}$ delivered from the KS, such as $C_{A,2} = (e(\alpha, rk_{B \rightarrow A}), \beta) = (Z^{a_2 \cdot b_1 \cdot k_i}, m \cdot Z^{b_1 \cdot k_i}) = (Z^{a_2 \cdot k'}, m \cdot Z^{k'})$ for $C_{B,r} = (\alpha, \beta)$. This enables both $\mathcal{A}$ and $\mathcal{B}$ to retrieve the encrypted block with their private keys, such as $m = m \cdot Z^{k_i} / (Z^{u_s \cdot k_i})^{u_s^{-1}}$ and $m = m \cdot Z^{u_1 \cdot k_i} / e(g^{k_i}, g)^{u_1}$, respectively, for $u \in \{a, b\}$ and $s \in \{1, 2\}$. This allows the storage of a single copy for duplicates with the aid of the trusted KS.

Stanek et al. [2013] presented deduplication of encrypted data that provides differential security based on the popularity of the outsourced data. The basic idea is to differentiate data protection depending on the popularity of the outsourced data by introducing convergent threshold encryption in combination with CE and a threshold encryption technique. Based on the observation that privacy-sensitive data is rarely shared by multiple data owners, repeatedly stored encryptions (i.e., unpopular data) of the same underlying content with different encryption keys are transformed into a single CE ciphertext (i.e., popular data) when the number of duplicates exceeds a predefined threshold. At the time when more than the predefined number of duplicates are about to be stored, the CSP interpolates the previously outsourced encryptions and transforms them into a single CE ciphertext.

Zheng et al. [2015] presented structure-aware deduplication tailored for video streaming to heterogeneous devices. Specific to multiple-layered video such as files encoded under scalable video coding (SVC), they introduced layer-level deduplication, which provides deduplication without destroying the underlying layer-based structure. In this scheme, the DupLESS approach is combined with ownership proof based on an equality test of hash values generated from common ciphertext. Duplicate checking for the base layer of the outsourcing file is performed via DupLESS, and then ownership proof is performed for each layer if there are duplicates.

5.1.3. Key Management. Deduplication exploits identical content by storing only a single instance of the duplicate data. For message-dependent encryption approaches, the same content is mapped to the same ciphertext for storage savings with or without aid of an additional KS. From another point of view, having a common key (namely, a convergent key) shared by multiple data owners for each outsourced data set means that data owners have to manage an enormous numbers of keys with the increasing amount of outsourced content. In answer to this, Li et al. [2014] developed distributed key management in deduplication, called *DeKey*, by introducing an independent master key for each data owner. The basic idea is to encrypt the encryption key using a

master key kept locally and securely by each data owner. A further enhancement to the basic approach is to leverage the secure management burden of the user master keys to multiple KSs by allowing the deduplication of convergent keys at a higher layer. In DeKey, as a combination of ramp SSS (RSSS) and a convergent dispersal algorithm, the authors presented convergent RSSS, which replaces random shares in RSSS with pseudorandom ones. (n, k, r)-RSSS is an SSS with n shares in which the secret can be restored for k shares and information leakage is restricted for less than r shares. During data outsourcing, n random shares in the modified RSSS are generated and distributed across the independent KSs only once by the data owner who first uploaded the data. This allows duplicate data owners to interact with a minimal number of KSs through authentication to reconstruct the complete encryption key from the secret shares (consequently, the plain data) received from the KSs. Specifically, by encoding $m = \lceil \frac{r}{k-r} \rceil$ pseudorandom shares of the secret convergent key $s = h(F)$ into n shares, in which m shares are generated using different hash functions such as $H_i = h(F + i)$ for $1 \leq i \leq n$, only valid data owners possessing the entire dataset are allowed to restore the complete convergent key as well as the original plain data. Koo et al. [2014] proposed another key management scheme for secure deduplication that considers dynamic updates of outsourced data. Whereas DeKey [Li et al. 2014a] and BDO-SD [Wen et al. 2015] conduct block-level deduplication only when file-level deduplication fails, file-level deduplication is performed for static data and block-level deduplication is performed for updated portions of the outsourced file along with corresponding metadata. Based on the inefficiency of a per-file encryption key in that each data owner has to manage an enormous number of keys with an increasing number of outsourced data and key exposure threats according to changes in the data-sharing group caused by data updates, per-user encryption with the key-encapsulation technique is adopted. Upon an update request for outsourced data, the outsourced encryption of the content and per-file public keys are updated at the CSP, whereas user-level private keys, kept by individual data owners, remain the same for every membership change in the data-sharing group owing to revocation, modification, and insertion. This reduces overhead in communication caused by the dynamic update of outsourced data as well as key management by data owners. Storer et al. [2008] presented block-level authenticated deduplication by means of the key-encapsulation technique in conjunction with authentication based on user identity. With the separation of KS from the CSP, the KS authenticates the data owner and provides both the encryption of the mapping information for the data using the KS's secret key and the encryption of the KS's secret key using the owner's public key. Upon access or upload request, using the data location and hash of the CE ciphertext from the duplicate data owner, the CSP provides the corresponding ciphertext to the owner after an identification test for the given hash value. From the viewpoint of data owners, this approach provides user revocation and eliminates the burden of key management, as the KS is given the power to authenticate and (re)encrypt mapping information using each user's public key. Similar to Storer et al.'s work, Puzio et al. [2013] proposed block-level deduplication with access control ability, called *ClouDedup*. They specified in detail the metadata required, such as file tables, pointer tables, and signature tables, with efficiency evaluated with respect to block size. This approach improves system performance by reducing bandwidth consumption and eliminating the re-encryption phase through the adoption of random storage. Wen et al. [2015] proposed deduplication supporting a keyword search, called *BDO-SD*, managing the encryption keys. Their main contribution is that although CE is adopted for data outsourcing with deduplication ability, a data owner queries the encrypted data with searchable encryption in a privacy-preserving manner. To do this, they combined an identity-based signature algorithm with a blind signature technique in a keyword search.

Zhou et al. [2015] employed user-aware convergent encryption (UACE) and multi-level key management (MLKM) in secure deduplication, which they called *SecDep*. Its primary characteristic is based on their observations that cross-user redundancies are mainly caused by duplicate files, whereas inside-user redundancies are caused by duplicate blocks. Therefore, to maximize the effectiveness of deduplication, both file-level and block-level encryption keys are generated with the aid of KSs and with the use of a user-aided method, respectively, leading to lower computational overhead. UACE combines cross-user file-level deduplication and inside-user chunk-level deduplication. In SecDep, MLKM uses a file-level key to encrypt block-level keys so that the key space does not increase with the number of sharing data owners. The MLKM splits the file-level keys into share-level keys via a Shamir SSS (SSSS) and delivers them to distributed KSs to ensure the security and reliability of file-level keys.

5.1.4. Comparative Analysis. This section analyzes the security and efficiency of the state-of-the-art deduplication schemes that exploit message-dependent encryption on the basis of the evaluation criteria presented in Section 4.2.

Efficiency. Each scheme has been evaluated with respect to system overheads for both clients and storage servers in terms of computation, storage, and communication costs. A comparison of efficiency for deduplication schemes based on these performance metrics is represented in Table I, where m is the number of chunks generated by splitting each file, n is the number of shares in SSS (i.e., the number of share holders), u is the number of data owners with the same hash value, and w is the number of keywords for the file when a keyword search is performed on the outsourced file. For the comparison of computational cost, H is the hash evaluation, Mul multiplication, Exp exponentiation, e the pairing operation, and Cmp the simple comparison operation. In addition, for well-known encryption/decryption techniques, SE is symmetric key encryption (e.g., AES) and PE is public key encryption (e.g., RSA or ElGamal). Otherwise, self-designed or customized encryption/decryptions are explained with primitive operations such as H, Mul, Exp, and/or e. Although iMLE provides the strongest security among the secure deduplication schemes, its computational overhead is omitted because the fundamental overhead for operations such as encryption and evaluation in fully homomorphic encryption appears excessive when compared to conventional cryptosystems and varies depending on the underlying assumptions (i.e., lattice, ring-LWE, integer, and NTRU).

Security. The security comparison is given in Table II. Satisfaction of the given criteria is illustrated using the codes ◯, △, ×, and –, which mean satisfactory, partially satisfactory, unsatisfactory, and not applicable, respectively. For confidentiality, if a scheme supports encryption and the security is bounded to the message distribution, it is marked as △. When a scheme supports encryption before outsourcing and its security is independent of the message distribution (i.e., secure against offline brute-force attacks), it is marked as ◯. Note that although Liu et al. [2014] and Storer et al. [2008] provide semantic security because keys are managed by a trusted KS, its confidentiality is based on stronger assumptions than those of the others, in which the KS cannot be fully trusted. Availability means resilience against failure or compromise of the KS so that the impact of a single point of failure is minimized for the entire deduplication system. None of the studies except Liu et al. [2015] provides side-channel resistance, as an adversary can identify at least the existence of duplicate data stored in cloud storage. Similar to side-channel resistance, ownership authenticity cannot be guaranteed in most approaches based on message-dependent encryption; the possession of the outsourced file is proven via a single small value as a proxy. The PoW technique,

Table I. Efficiency Comparison for Message-Dependent Encryption Schemes

	Schemes	Computation		Storage		Communication
		Client	Server	Client	Server	
Client-side encryption	MLE [Bellare et al. 2013a]	$O(1)H + O(1)SE$	–	$O(1)$	$O(1)$	$O(1)$
	Abadi et al. [2013]	$O(n)H + O(n)Mul + O(n)Exp + O(1)SE + O(n)PE$	$O(n)H + O(1)e$	$O(1)$	$O(n)$	$O(1)$
	Liu et al. [2015]	$O(u)H + O(u)Exp + O(1)SE$	$O(u)Cmp$	$O(1)$	$O(1)$	$O(u)$
Server-aided encryption	DupLESS [Bellare et al. 2013b]	$O(1)H + O(1)Mul + O(1)Exp + O(1)SE$	$O(1)Exp$	$O(1)$	$O(1)$	$O(1)$
	Miao et al. [2015]	$O(1)H + O(n)Mul + O(n)Exp + O(1)SE$	$O(n)Exp$	$O(1)$	$O(1)$	$O(n)$
	Duan [2014]	$O(1)H + O(n)Mul + O(n)Exp + O(1)SE$	$O(1)Exp$	$O(1)$	$O(1)$	$O(n)$
	Li et al. [2015a]	$O(1)H + O(1)SE$	$O(1)H$	$O(1)$	$O(1)$	$O(m \log m)$
	Liu et al. [2014]	$O(m)H + O(m)Mul + O(m)Exp + O(m)SE + O(m)e$	$O(m)Mul + O(m)Exp + O(m)e$	$O(1)$	$O(m)$	$O(1)$
	Stanek et al. [2013]	$O(1)H + O(1)SE + O(1)e$	$O(n)Exp$	$O(1)$	$O(n)$	$O(n)$
	Zheng et al. [2015]	$O(1)H + O(1)Exp + O(1)SE$	$O(1)H$	$O(1)$	$O(1)$	$O(1)$
Key management	Li et al. [2014]	$O(m)H + O(m)Mul + O(m)Exp + O(m)SE$	$O(m)Exp + O(1)e$	$O(1)$	$O(m)$	$O(1)$
	Koo et al. [2014]	$O(1)H + O(1)Mul + O(m)Exp + O(m)e$	$O(m)Exp$	$O(1)$	$O(m)$	$O(1)$
	Storer et al. [2008]	$O(1)H + O(m)SE$	$O(m)SE$	$O(1)$	$O(m)$	$O(1)$
	ClouDedup [Puzio et al. 2013]	$O(m)H + O(m)SE$	$O(m)SE$	$O(1)$	$O(m)$	$O(m \log m)$
	BDO-SD [Wen et al. 2015]	$O(1)H + O(1)Mul + O(m)Exp + O(m)SE$	$O(m)Cmp$	$O(m)$	$O(m)$	$O(m)$
	SecDep [Zhou et al. 2015]	$O(m)H + O(m+n)Mul + O(m+n)Exp + O(m)SE$	$O(m)Cmp$	$O(m)$	$O(m)$	$O(n)$

Table II. Security Comparison for Message-Dependent Encryption Schemes

	Schemes	Confidentiality	Availability	Side-Channel Resistance	Ownership Authenticity	Data Integrity
Client-side encryption	MLE [Bellare et al. 2013a]	△	×	×	×	◯
	Abadi et al. [2013]	◯	×	×	×	◯
	iMLE [Bellare and Keelveedhi 2015]	◯	×	×	×	◯
	Liu et al. [2015]	◯	◯	–	×	×
Server-aided encryption	DupLESS [Bellare et al. 2013b]	◯	×	×	×	◯
	Miao et al. [2015]	◯	◯	×	×	×
	Duan [2014]	◯	◯	×	×	×
	Li et al. [2015a]	◯	×	×	×	×
	Liu et al. [2014]	◯	×	×	×	×
	Stanek et al. [2013]	△	×	×	×	×
	Zheng et al. [2015]	◯	×	×	×	×
Key management	Li et al. [2014]	△	◯	×	◯	◯
	Koo et al. [2014]	△	×	×	×	◯
	Storer et al. [2008]	◯	×	×	×	×
	ClouDedup [Puzio et al. 2013]	◯	◯	×	×	×
	BDO-SD [Wen et al. 2015]	◯	◯	×	◯	◯
	SecDep [Zhou et al. 2015]	◯	◯	×	×	◯

described in Section 5.2, is adopted to provide ownership authenticity only in DeKey [Li et al. 2014a] and BDO-SD [Wen et al. 2015].

Deduplication schemes based on client-side encryption generally suffer from dictionary attacks on predictable message space, as adversaries can identify the existence of an outsourced file through a brute-force attack on predictable messages. Most server-aided encryption schemes attempt to mitigate this identification by adversaries, and encryption is performed with the aid of a KS, which incurs additional communication overhead and leverages computational overhead to the server. However, server-aided encryption cannot completely eliminate dictionary attacks for cross-user deduplication, as it still reveals the existence of an outsourced file. Deduplication schemes supporting key management mechanisms provide secure key management and update outsourced content to address accidental information breaches. Such schemes adopt bilinear maps to update the outsourced content stored outside their premises, causing computationally intensive operations on the client side.

5.2. Proof of Ownership

Cloud storage systems with client-side deduplication face a new kind of security threat stemming from the fact that in typical client-side deduplication, a very small fingerprint, such as a hash signature, calculated from the data is used to identify the existence of the same data in the storage. With knowledge of this hash value, an adversary who does not have the data can convince the CSP that it owns the data just by presenting the hash value, hence enabling the download of the data from the cloud storage.

To mitigate this threat, PoW was introduced. A PoW scheme is a challenge-response protocol that allows a CSP to verify a client's ownership for particular data. In the protocol, a client should prove to a CSP that it actually holds the entire set of data rather than just the hash value.

Generally, a PoW is composed of a summary function S and an interactive two-party protocol $\Pi = (P, V)$ between prover P (a client) and verifier V (a CSP) on joint input data F. Specifically, the two-party protocol Π includes four main steps: initialization, challenge, proof, and verification. In *initialization*, the verifier generates short information v for F using summary function S. In *challenge*, to check whether the prover owns F, the verifier randomly selects challenges and sends them to the prover. In *proof*, upon receiving the challenges, the prover generates a short proof

for the received challenges and responds with the proof. In *verification*, the verifier determines if the received proof is valid with v and the challenges.

If the probability that an adversary without F can deceive the CSP by proving its ownership is negligible given a security parameter, then the PoW is considered to be secure even if part of the data is leaked to the client. This security model actually can be seen as an example of a bounded retrieval model [Dziembowski 2006]. In other words, as long as the min-entropy in the data (from the adversary's perspective) is more than the sum of the number of bits given to the adversary and the security parameter (in bits), the adversary should not be able to convince the CSP that it possesses the data with nonnegligible probability.

The efficiency of PoW schemes can be measured in terms of computation, I/O, and network bandwidth on both the client's and server's sides. Additionally, PoW schemes should avoid requiring the server (or the CSP) to load large amounts of the data from backend storage for each execution of the protocol.

In the rest of this section, we categorize the existing PoW schemes into MHT-based PoW, spot checking–based PoW, and auditable PoW approaches. They are then reviewed and analyzed in terms of their advantages and disadvantages.

5.2.1. MHT-Based PoW. Several PoW approaches have been constructed based on the MHT [Merkle 1990]. MHT is a binary tree in which every leaf node holds a block of data and every non-leaf node is labeled with a hash value of the labels of its children nodes. In MHT-based PoW schemes, once the entire data file is received from a client, the CSP encodes it using an erasure code, meaning that it is able to recover the entire data file from any part of the encoded bits. The CSP then builds an MHT over the encoded data and generates shorter verification information. Later, any clients attempting to outsource the data will run the PoW protocol with the CSP to prove that it really has the data. Interacting with the client, the CSP verifies materials that are sent from the client with the verification information generated from the data.

Halevi et al. [2011] first proposed the concept of MHT-based PoW and presented three different constructions trading off security and efficiency. These constructions are built on a common approach, and thus we only present the first construction in Halevi et al.'s scheme in this section. The construction applies erasure coding to the content of the data. Let $E : \{0, 1\}^M \rightarrow \{0, 1\}^{M'}$ be an α-erasure code, which is able to recover up to an α fraction of corrupted bits, and let H be a collision resistant cryptographic hash function. An MHT over data buffer X with leaves of b bits size and the hash function H is denoted as $MT_{H,b}(X)$. For input data $F \in \{0, 1\}^M$ of M bits, the verifier (i.e., the CSP) generates the encoding $X = E(F)$ as well as $MT_{H,b}(X)$, then stores the root value of the tree as the verification information for F. When a proof protocol is initiated, the verifier chooses at random sufficient leaves, $l_1, l_2, \ldots, l_u$, where u depends on the security parameter, and sends the indexes of these leaves to the prover (i.e., the client). The prover then responds with the sibling-paths of all of these leaves (i.e., a set of the siblings of the nodes on the path from a leaf to the root) to the verifier. After verifying the prover's response with respect to $MT_{H,b}(X)$, the verifier returns to the prover Accept, if it is valid, or Fail, if otherwise. In Halevi et al.'s scheme, the CSP is assumed to be trusted since data blocks at the challenged leaves are fully revealed to the CSP during protocol execution. Hence, this scheme is appropriate for private cloud storage services in which storage servers are co-located with its users in the same organization.

Unfortunately, PoW is sometimes required over private data in a public cloud where the CSP cannot be fully trusted. In this situation, a client who owns private data should be able to prove to the CSP that it actually has the entire dataset without revealing further information to the CSP. A simple way of achieving this goal is combining the aforementioned PoW scheme with encryption techniques. In this approach, however,

applying a semantically secure encryption algorithm to data blocks on the leaf nodes of an MHT would randomize the root value, thus making it impossible to identify the same dataset on the cloud server.

Ng et al. [2012] challenged this problem and proposed a private PoW scheme over encrypted data, building upon Halevi et al.'s approach. Let us denote i-th data block B_i with m symbols as $B_i = (B_{i,1}, \ldots, B_{i,m})$. Instead of a data block, each leaf node in an MHT stores the commitment of block $C_i = g_1^{B_{i,1}} \cdots g_m^{B_{i,m}}$, where $g_1, \ldots, g_m$ are m generators of cyclic group $\mathbb{G}$ with prime order q. Upon receipt of the index of a leaf node as a challenge message, the client responds with the sibling path from the selected index as well as the corresponding commitment C_i. The client and the CSP then initiate a zero-knowledge proof–based two-party computation protocol. By executing the protocol, the CSP can verify the client's ownership of the file based on the material presented by the client without knowing any information within the file. Ng et al. proved that this is secure under the assumption that solving the discrete logarithm problem (DLP) is infeasible and the underlying hash algorithm is collision resistant. In terms of computation efficiency, however, this scheme incurs high computation costs, requiring the generation of all commitments by modular exponentiation.

Another solution to the PoW problem over private data can utilize the CE technique that always generates the same ciphertext deterministically, as described in the previous section. In the bounded leakage model of general PoW, however, an encryption key for CE is treated as insecure and assumed to be easily leaked to adversaries since the key is generated from input data in a deterministic way. Xu et al. [2011, 2013] proposed another encryption method similar to CE but employing a randomized encryption key. Their encryption method generates two ciphertexts (C_F, C_τ): C_F is a ciphertext of a data file F computed with AES and random key τ, and C_τ is generated by encrypting AES key τ with the input data as an encryption key. They also constructed nonlinear and keyed pairwise-independent hash function $H_K(\cdot)$ in a random oracle model, and applied it to produce a hash of data $H_K(F)$ on which an MHT is built. A client who initially uploads F will send the ciphertexts (C_F, C_τ) and root value v_r of MHT over $H_K(F)$ to the CSP. Any subsequent uploading client will prove his or her ownership of F by running an MHT-based PoW protocol over $H_K(F)$.

Kaaniche and Laurent [2014] also proposed an MHT-based PoW scheme over encrypted data, and they validated its effectiveness by implementing it in a real storage framework based on an open storage system (Swift). This method adopts CE to protect data confidentiality against an untrustworthy CSP, which leads to a weakness in security in the bounded leakage model.

5.2.2. Spot Checking–Based PoW. The MHT-based PoW solutions described in the previous section commonly suffer from several shortcomings that might hinder their adoption by cloud storage services. The main drawback is that they require high I/O and computation overheads on the client side. For each protocol execution, a client has to load the entire data from disk to a memory buffer and then build an MHT in the buffer, which usually requires a number of I/Os and hash computations. Another drawback is that their security depends on an assumption that is hard to analyze.

To overcome these drawbacks, several studies have adopted a spot checking–based method as a PoW protocol. In this approach, a CSP sends randomly selected positions within a data file in bits or blocks to a client as a challenge message. Then, without any precomputations such as building a Merkle tree, the client responds with a proof that just consists of the bits or blocks corresponding to the challenged positions. In this way, a client is allowed to avoid heavy computation or numerous I/O operations to prove his or her ownership of a particular data file. The security of this approach can be proven information theoretically and analyzed more easily compared to MHT-based methods.

Efficiency gain is achieved on the client side; however, an extra burden may fall on the CSP because the verification of the proof sent from a client demands access to data stored in large backend storage, which usually leads to increased I/O costs.

Di Pietro and Sorniotti [2012] proposed s-PoW, the first spot checking–based PoW solution. The authors mainly addressed the inefficiency of the MHT-based PoW scheme and proposed an efficiency-enhanced version of the PoW protocol in terms of computational power and I/O capacity. The basic idea of s-PoW comes from the fact that an adversary's probability of success in learning some portion of K bits of data is negligible in the security parameter. In the s-PoW scheme, pseudorandom number generator $PRG(s)$ is given to the clients and CSP during the initialization phase. $PRG(s)$ takes s as a seed and generates an integer in [1,M], where M is the size of the data in bits. For each challenge, a new random seed c is chosen by the CSP and sent to the client. Upon receipt of the challenge, the client selects K random positions using PRG and constructs a K-bit response string by concatenating each bit at each selected position in the data. The CSP can verify the response from the client by comparing bits in the message to that of the data in the storage. To avoid frequent and expensive access to backend storage for verification, s-PoW uses a precomputation approach. In other words, when a data file is initially uploaded, the CSP generates tuples of a challenge and its corresponding response message in advance, and uses them for later protocol executions. After all of the precomputed challenges have been used, the CSP will generate other tuples by accessing the data in the storage.

Regarding the enhancement of efficiency in terms of I/O complexity for the CSP, Blasco et al. [2014] utilized a Bloom filter and applied it to the construction of a new PoW protocol—bf-PoW. A Bloom filter is a relatively small set of data elements that is used to perform probabilistic membership queries over them. In the bf-PoW scheme, for each uploaded data, a Bloom filter is built by the CSP and kept in the storage to be used later to verify a client's challenges. A Bloom filter can be created as follows. In the initialization phase, the CSP splits the data into chunks of equal size and generates tokens (i.e., hashes) for each chunk. The CSP then evaluates pseudorandom function PRF using the tokens as seeds and inserts the outputs of PRF into the Bloom filter. The CSP will start the challenging phase by sending some of the indexes of chunks selected randomly to a client. Upon receiving the challenge message, the client responds with the tokens for the challenged chunks. The CSP converts the received tokens into PRF outputs and checks the membership of the outputs in the Bloom filter. If all of the membership tests for the tokens are passed, then verification is considered to be successful. Otherwise, the verification is considered to have failed. In the bf-PoW scheme, the CSP is only required to load the Bloom filter for a particular file into the main memory, not the entire content of the data. Hence, bf-PoW is more efficient than s-Pow in terms of I/O cost on the server side.

González-Manzano and Orfila [2015] considered the problem of protecting data confidentiality against honest-but-curious CSPs, which is not addressed in s-PoW and bf-PoW. Their proposal, dubbed ce-PoW, is a spot checking–based method combined with a CE technique. Because applying CE directly to data in the bounded leakage setting is not secure [Xu et al. 2011], ce-PoW decomposes the data into a number of chunks and then performs CE on each chunk rather than on the data as a whole. The number of chunks will be determined according to the size of the data so that the total size of the convergent keys is not less than the leakage bound. ce-PoW also provides a mechanism in which a CSP can check the consistency between a convergently encrypted chunk and its tag, which can eventually mitigate poisoning attacks. Regarding complexity, the way that a CSP handles an interaction with a client follows the approach of s-PoW. In other words, it uses precomputed tuples of challenges and corresponding responses.

In some situations, by utilizing pairs of challenges and responses collected over a number of protocol instances that were previously exchanged between a CSP and

authorized clients, it might be possible for an adversary to fool the CSP with high probability. To ensure security against this type of threat, Yang et al. proposed two PoW solutions—POF [Yang et al. 2013a, 2013b] and POEF [Yang et al. 2015]. The POF scheme uses a dynamic spot checking–based approach in which for every challenging phase, the CSP randomly chooses not only the indexes of the challenge blocks but also their coefficients. Randomizing the coefficients can guarantee the freshness of the proof messages in every protocol instance and make it impossible for an adversary to launch an attack. POEF is a variant of POF that enables POF over convergently encrypted data. In terms of complexity, these two schemes suffer from high I/O because for every challenge, the corresponding data blocks need to be accessed from backend cloud storage for verification.

5.2.3. PoW with Auditability. In addition to deduplication security, data integrity is also crucial to cloud storage systems. Proof of data possession (PDP) [Ateniese et al. 2007] and proof of retrievability (POR) [Juels and Kaliski Jr. 2007] were introduced to allow data owners or auditors to remotely check whether outsourced data has been tampered with or not. To achieve integrity and security while not losing storage efficiency, there have been some efforts to construct a PoW scheme that offers data auditability by combining PoW with PDP/POR.

Zheng and Xu [2012] proposed proof of Storage with Deduplication (POSD). The main approach takes advantage of public verifiability, which is a property of PDP/POR in which an auditor can verify the integrity of outsourced data without knowing any secret information about the data. By reversing the roles of the auditor (or a client) and the CSP in a PDP/POR construction, a PoW protocol can be easily implemented. In the POSD scheme, each client sets up his or her own public/private key in the initialization phase and creates an authentication tag for each data by using the key pair. The data and its authentication tag will then be outsourced together to a CSP, and the data owner's public key will be made public. In the case of data auditing, any auditors and data owners holding a public key can ensure the integrity of the data by verifying the proof generated by the server over the data and its corresponding tags. On the other hand, using the initial uploader's public key and the authentication tag, the CSP can verify the ownership of a challenged client.

Shin et al. [2012] identified a security vulnerability in POSD and revealed the existence of *weak-key* attacks on the scheme. This security weakness arises from the fact that a verification algorithm depends on client-supplied keys, and thus the validity of the protocol stands on the strong assumption that all clients honestly generate their keys. Shin et al. proposed e-POSD, which enhances the security of POSD by minimizing the key generation capability of clients. In other words, the CSP blends the client public key with random values when the data and corresponding tags are received.

The POSD scheme also suffers from significant computational and communication costs that increase linearly with the number of clients and the size of the data. Yuan and Yu [2013] proposed PCAD, a new solution to address the efficiency issues of the POSD scheme. PCAD is based on polynomial-based authentication tags and homomorphic linear authenticators. Because it aggregates multiple authentication tags into a single message, the total computational and communication cost of PCAD remains constant regardless of the number of data owners or the size of outsourcing data.

5.2.4. Comparative Analysis. The PoW schemes are compared with respect to efficiency and security on the basis of the evaluation criteria presented in Section 4.2.

Efficiency. Each scheme has been evaluated for both the client and CSP (i.e., the cloud storage server) in terms of computation, storage, and communication costs. The efficiency analysis and comparison results for PoW schemes are presented in Table III, where m is the number of blocks in each data; s is the size of blocks (i.e., the number of

Table III. Efficiency Comparison for PoW Schemes

	Schemes	Computation		Storage		Communication
		Client	Server	Client	Server	
MHT	Halevi et al. [2011]	$O(m)H$	$O(\log k)H$	$O(1)$	$O(1)$	$O(k \log k)$
	Ng et al. [2012]	$O(m)Exp$	$O(\log k)H$	$O(1)$	$O(1)$	$O(k \log k)$
	Xu et al. [2013]	$O(m)H + O(m)Enc$	$O(m)H$	$O(1)$	$O(m)$	$O(1)$
	Kaaniche and Laurent [2014]	$O(m)H + O(m)Enc$	$O(\log k)H$	$O(1)$	$O(1)$	$O(k \log k)$
Spot checking	s-PoW [Di Pietro and Sorniotti 2012]	0	$O(pk)H$	$O(1)$	$O(pk)$	$O(k)$
	bf-PoW [Blasco et al. 2014]	$O(k)H$	$O(k \log(1/r_f)/r_f)H$	$O(1)$	$O(\log 1/r_f)$	$O(k/r_f)$
	ce-PoW [González-Manzano and Orfila 2015]	$O(k)Enc$	$O(pk)H$	$O(1)$	$O(pk)$	$O(k)$
	POF [Yang et al. 2013b]	$O(k)H$	$O(k)H$	$O(1)$	$O(m)$	$O(1)$
	POEF [Yang et al. 2015]	$O(k)Enc$	$O(k)H$	$O(1)$	$O(m)$	$O(1)$
PoW w/ audit.	POSD [Zheng and Xu 2012]	$O(sk)Mul$	$O(sk)Mul + O(k)Exp$	$O(1)$	$O(m)$	$O(s)$
	e-POSD [Shin et al. 2012]	$O(sk)Mul$	$O(sk)Mul + O(k)Exp$	$O(1)$	$O(m)$	$O(s)$
	PCAD [Yuan and Yu 2013]	0	$O(s)Mul + O(s)Exp$	$O(1)$	$O(m)$	$O(1)$

Table IV. Security Comparison for PoW Schemes

	Schemes	Confidentiality	Availability	Side-Channel Resistance	Ownership Authenticity	Data Integrity
MHT	Halevi et al. [2011]	×	×	×	◯	×
	Ng et al. [2012]	△	×	×	◯	×
	Xu et al. [2013]	△	×	×	◯	×
	Kaaniche and Laurent [2014]	△	×	×	◯	×
Spot checking	s-PoW [Di Pietro and Sorniotti 2012]	×	×	×	◯	×
	bf-PoW [Blasco et al. 2014]	×	×	×	◯	×
	ce-PoW [González-Manzano and Orfila 2015]	△	×	×	◯	×
	POF [Yang et al. 2013b]	×	×	×	◯	×
	POEF [Yang et al. 2015]	△	×	×	◯	×
PoW w/ audit.	POSD [Zheng and Xu 2012]	×	×	×	△	△
	e-POSD [Shin et al. 2012]	×	×	×	◯	◯
	PCAD [Yuan and Yu 2013]	×	×	×	◯	◯

group elements in each block); k is the security parameter; r_f is the false-positive rate of the Bloom filter; p is the number of precomputed challenges; H is a hash (or PRF) computation; Mul and Exp are group multiplication and exponentiation, respectively; and Enc is an encryption. MHT-based PoW schemes generally suffer from very high computation costs on the client side because the construction of an MHT on the outsourced data should be performed prior to the execution of protocol. Spot checking–based PoW schemes simply adopt the random sampling of data blocks instead of building an MHT, thus greatly reducing computational overhead for the client. However, the CSP has to directly access the content of challenged blocks to verify response messages, which leads to significant I/O demands in the backend storage for each instance of the protocol. PoW schemes with auditability (POSD, e-POSD, and PCAD) have the goal of auditing the outsourced data in addition to proving data ownership. To achieve auditability, these schemes utilize group multiplication and exponentiation, which lead to increased computational costs for the CSP.

Security. Table IV displays the results of the security comparison for the PoW schemes. Since confirming the ownership of the outsourced data for valid clients in client-side deduplication systems is the main concern of PoW-based approaches, ownership authenticity is satisfied by all schemes. Some solutions—POSD [Zheng and Xu 2012], e-POSD [Shin et al. 2012], and PCAD [Yuan and Yu 2013]—also consider the verification of the integrity of the outsourced data. However, as identified by Shin et al. [2012], ownership authenticity and data integrity in POSD is susceptible to weak-key

attacks. Furthermore, several MHT-based and spot checking–based schemes [Ng et al. 2012; Xu et al. 2013; Kaaniche and Laurent 2014; González-Manzano and Orfila 2015; Yang et al. 2015] provide solutions for the confidentiality of outsourced data. Compared to the server-aided encryption schemes in Section 5.1.2, the security of these schemes in terms of confidentiality is weak. This is because data confidentiality can be guaranteed only if the message distribution is large and unpredictable.

5.3. Traffic Obfuscation

Client-side deduplication architecture is also vulnerable to the threat of outside adversaries learning sensitive information through side channels. In other words, by using observed information such as the volume of network traffic transmitted during the duplication protocol, an adversary can easily determine whether a certain file exists on a remote server or not, and even infer its content. To prevent adversaries from utilizing side channels during deduplication, several traffic obfuscation solutions have been proposed that make network traffic look randomized so that adversaries are unable to infer any information by monitoring the network.

5.3.1. Server-Based Obfuscation. Harnik et al. [2010] focused on weakening the correlation between deduplication and the existence of files in storage, proposing a randomized approach that obfuscates the occurrence of deduplication. In this protocol, the CSP assigns threshold t_F to each file F. The threshold is chosen randomly from range $[2, d]$, where d is a public security parameter. The CSP records the number of previous uploads of F as counter c_F. When a client wants to outsource a new copy of F, the CSP checks whether the accumulated number of file uploads c_F exceeds the threshold t_F. If $c_F \geq t_F$, the CSP performs data deduplication on the client side and a copy of that file is not sent over the network. Otherwise, deduplication will be performed on the server side, so the file has to be sent to the CSP.

The approach of Harnik et al. [2010] might impose some communication overhead since for the first $t_F - 1$ uploads of F, the whole file should be sent over the network. After F has been uploaded, however, the CSP can do a duplicate check on F and then run deduplication. Regarding security, Harnik et al.'s scheme does not allow an adversary to distinguish between cases where a single copy of F has already been uploaded and those where the file was not uploaded.

Lee and Choi [2012] observed that the success probability of an attack against the scheme of Harnik et al. [2010] (i.e., $1/d-1$) is nonnegligible in a security parameter, and thus user privacy is still at risk. Following this observation, they proposed a security-enhanced version of Harnik et al.'s solution. In Lee et al.'s scheme, for every file F, threshold t_F is initially set to d, where d is a security parameter. When a copy of F has been initially uploaded, the CSP chooses $r \in \{0, 1, 2\}$ at random and computes $t_F = t_F - r$ instantly. Client-side deduplication then occurs if $c_F \geq t_F$ or the copy of F is uploaded by the same user; otherwise, server-side deduplication will be performed. As analyzed in Lee and Choi [2012], the success probability of an attack can be minimized when the security parameter d is chosen to be a multiple of 3. The probability of an adversary learning information about the existence of F is $(\frac{1}{3})^{d/3} + (\frac{1}{3})^{d}$.

5.3.2. Gateway-Based Obfuscation. The drawback of server-based traffic obfuscation is that large volumes of network traffic between the client and the CSP are inevitable even when duplicate uploads are found. In certain network environments, such as a wide area network (WAN) or the Internet, where transmission costs are high, the waste of bandwidth due to obfuscation is unsustainable. Gateway-based traffic obfuscation is an alternative solution for these types of network. In certain cloud service models including hybrid cloud computing, a (storage) gateway can be deployed at the boundary

Table V. Efficiency Comparison for Traffic Obfuscation Schemes

	Schemes	Computation		Storage		Communication
		Client	Server	Client	Server	
Server-based obfuscation	Harnik et al. [2010]	–	–	–	–	$B = d/2$
	Lee and Choi [2012]	–	–	–	–	$B = d/2$
Gateway-based obfuscation	Heen et al. [2012]	–	–	–	–	$B = 0$ * Home router is required.
	Shin and Kim [2015]	–	–	–	–	$B = 0$ * Storage gateway is required.

B: average number of uploading a duplicate copy of a file (bandwidth penalty)

between a WAN and the customer's on-premise network, and provide clients, located in a local area network (LAN), with an interface to cloud storage services. A gateway is usually an appliance server with a large disk and considered to be trusted from the perspective of both the CSP and the client. Executing an obfuscation algorithm at the gateway rather than at the CSP (or the cloud storage server) offers benefits in terms of network utilization while not losing the required security.

Heen et al. [2012] proposed a gateway-based solution suitable for a specific service deployment model in which a network service provider offers a cloud storage service through a gateway device that resides on the customer's local home network. A home gateway is located at the border between a LAN, an insecure zone that is likely to be compromised by an adversary, and a WAN or the Internet, a secure zone that is unlikely to be controlled by an adversary. When a user requests to upload a particular file to cloud storage, the home gateway (the gateway server module) handles this request and receives the entire file on behalf of the server. Upon receipt of the file, the gateway, in the role of the client, performs the deduplication protocol with the CSP and uploads the file to the CSP if necessary. Since the deduplication operation always happens only on the WAN (i.e., the secure zone) and the bandwidth management module in the gateway mixes the traffic of deduplication with that of other services, it is impossible for an adversary to learn information about an interesting file by monitoring network traffic.

The restriction to a specific service model is the main disadvantage of the method of Heen et al. [2012]. Shin and Kim [2015] proposed another solution that is more suitable for a hybrid cloud computing model, which is flexible and has been widely used in various types of organizations. The authors also identified a plausible type of attack, a *related-files* attack, on server-based obfuscation. This attack exploits the cohesion that exists among a group of files outsourced together (i.e, files with identical content but different document formats). By learning information through side channels about not only an interesting file but also other related files, the probability of success for the adversary will greatly increase. To mitigate the attack, Shin and Kim applied a differentially private mechanism [Dwork 2008] to a gateway-based obfuscation algorithm. Differential privacy guarantees that it is infeasible to learn of the presence or absence of a particular file in remote storage through side-channel attacks.

5.3.3. Comparative Analysis.

Efficiency. Traffic obfuscation schemes inevitably generate dummy network traffic according to their obfuscation algorithms. Thus, communication costs over the network is the main concern when analyzing their performance. Table V shows the comparison between traffic obfuscation schemes. To compare communication costs, the bandwidth penalty (B) is defined as the average number of file uploads even if the duplicate copy is found. For each instance of the deduplication protocol, server-based obfuscation schemes randomly choose between server-side and client-side deduplication. The probability that server-side deduplication is chosen is $d/2$, where d is the public security parameter. Hence, the bandwidth penalty for server-based obfuscation schemes is

Table VI. Security Comparison for Traffic Obfuscation Schemes

	Schemes	Confidentiality	Availability	Side-Channel Resistance	Ownership Authenticity	Data Integrity
Server-based obfuscation	Harnik et al. [2010]	×	×	△ ($p = 2/d - 1$)	×	×
	Lee and Choi [2012]	×	×	△ ($p = 1/3^{d/3} + 1/3^{d}$)	×	×
Gateway-based obfuscation	Heen et al. [2012]	×	×	◯	×	×
	Shin and Kim [2015]	×	×	◯	×	×

p: probability of information leakage

$B = d/2$. In gateway-based obfuscation schemes, deduplication is obfuscated by mixing other traffic with the aid of a gateway, so duplicate file uploads do not occur (i.e., the bandwidth penalty is $B = 0$).

Security. Table VI shows the security comparison between traffic obfuscation schemes. The scheme of Harnik et al. [2010] and that of Lee and Choi [2012] offer side-channel resistance against outside adversaries. However, with some probability, information about the outsourced file can be leaked to the adversary through side channels. The information leakage probability (p) depends on the public security parameter d. They are also vulnerable to related-file attacks. Gateway-based obfuscation schemes are secure against related-file attacks and provide stronger security than do server-based schemes. However, they require the assumption that the gateway is not compromised by adversaries.

5.4. Deterministic Information Dispersal

Outsourcing data to a single cloud storage server is vulnerable to the single point of failure problem, which can lead to data corruption. Utilizing data deduplication techniques further raises concerns about the availability of outsourced data due to the nature of deduplication, which deletes all redundant data and maintains only a single copy regardless of the number of data owners.

To address the availability issue, the research community has investigated a decentralization approach that exploits multiple CSPs. Multicloud storage combines two or more public cloud services into a single storage service and guarantees data availability using an IDA, which transforms data into multiple shares with some redundancy and disperses the shares across multiple CSPs. Certain IDAs, such as secret sharing [Shamir 1979], provide further data confidentiality even if a subset of shares are leaked. However, the existing dispersal algorithms rely on a random value embedded in the shares to maintain the security, which eventually prohibits deduplication of the dispersed data. Some proposals have overcome this problem by constructing a deterministic IDA. Similar to CE, the idea is to replace random information in the original algorithm with deterministic information such as a hash value derived from the data. We describe existing deterministic IDAs in detail.

5.4.1. Deterministic IDA. In its formal definition, an (n, k, r)-dispersal algorithm, where $n > k > r \geq 0$, disperses secret data into n shares such that the original secret can be recovered from any k shares and information about the secret cannot be learned from any r shares. Li et al. [2014b] first proposed two IDAs—CRSSS and CAONT-RS—that enable the deduplication of the dispersed shares.

CRSSS is constructed from the RSSS [Blakley and Meadows 1985] by modifying the algorithm in a deterministic way. Given a secret, CRSSS first splits it into a number of words with the same size as hashes (e.g., the size of word is set to be 32 bytes if SHA-256 is used). For each group of $k - r$ words in the secret, the algorithm computes r hashes from the $k - r$ words. To ensure security, each hash value h_i is generated using

a different cryptographic hash function as follows:

$$h_i = H(D||i), \text{ for } i = 0, 1, \ldots, r-1,$$

where D refers to the block composed of $k-r$ words, $||$ denotes concatenation, and H is a cryptographic hash algorithm such as SHA-256. The k words, which consists of the $k-r$ words together with the computed r hashes, are then transformed into n shares using Rabin's IDA [Rabin 1989]. In CRSSS, it can be shown that unless all $k-r$ words are collected, the adversary cannot infer any information on any of the r hashes. Deterministic hash values are used instead of random information, and thus deduplication is available for the shares generated by CRSSS.

CAONT-RS is based on the all-or-nothing transform with Reed-Solomon coding (AONT-RS) [Resch et al. 2011] algorithm and is composed of two phases: convergent AONT (CAONT) and Reed-Solomon coding (RS). In the CAONT phase, the algorithm splits a secret into a group of s words with the same size as the hashes. The s words, $d_0, d_1, \ldots, d_{s-1}$, are then transformed into $s+1$ CAONT words, $c_0, c_1, \ldots, c_s$, as follows:

$$c_i = d_i \oplus E(h_{key}, i), \text{ for } i = 0, 1, \ldots, s-1$$
$$c_s = h_{key} \oplus H(c_0||c_1|| \ldots ||c_{s-1}),$$

where h_{key} is the hash that is generated from the secret, E is a symmetric encryption algorithm, and H is a cryptographic hash algorithm. In the RS phase, the $s+1$ CAONT words are treated as a CAONT package, and they are divided into k shares. Finally, the algorithm outputs n shares by encoding the k shares using a systematic RS code. Due to the underlying security property of AONT, h_{key} cannot be inferred unless the entire CAONT package is given to an adversary. As in CRSSS, there is no random information in CAONT-RS, and thus the algorithm enables deduplication on the dispersed data.

Li et al. [2015b] further extended their work by implementing CDStore, a unified multicloud backup solution using the proposed convergent IDA. Being deployed across commercial CSPs such as Google, Microsoft Azure, and Rackspace, CDStore realizes the concept of decentralized storage in which the properties of fault tolerance and reliability can be preserved. In their solution, each user runs a CDStore client program to outsource and access their data in multiple clouds over the Internet. The client program provides to the user a virtual view of a single storage, but actually CAONT-RS divides the outsourced data into a number of shares and disperses the shares across multiple CSPs. A CDStore server, which runs in each cloud as a virtual machine, handles the data shares uploaded by users and performs deduplication of the data. By evaluating real-world datasets under various workloads, Li et al. validated the effectiveness of the multicloud approach as well as their convergent dispersal algorithm.

Another deduplication solution using decentralized servers was proposed by Li et al. [2015]. In their proposal, a deterministic dispersal algorithm was constructed based on the SSSS [Shamir 1979]. As in the approach used by Li et al. [2015b], SSSS is modified to become a deterministic algorithm as follows. Given secret a_0, a $(k-1)$-degree polynomial $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1} \in Z_p[X]$ is picked by computing the coefficients from the secret such that $a_i = H(a_0||i)$, for $i = 1, \ldots, k-1$, where p is a prime and H is a cryptographic hash function. Each share $\sigma_i = f(i)$ $(1 \leq i \leq n)$ is then outsourced to the i-th CSP. Due to its deterministic property, shares generated from data with a small message space may be predictable and vulnerable to guessing attacks. To prevent adversaries from guessing, the authors also presented an enhanced version of the scheme that adds unpredictability to the data shares by employing the oblivious pseudorandom function (OPRF) protocol, which is similar to DupLESS [Bellare et al. 2013b].

The deterministic secret sharing algorithm of Li et al. [2015b] outputs n shares given a single secret (i.e., $k = 1$), and thus the storage blowup, which is the ratio of the size of

Table VII. Efficiency Comparison for Information Dispersal Schemes

Schemes	Computation		Storage		Communication
	Client	Server	Client	Server	
CRSSS [Li et al. 2014a]	–	–	–	$\rho = n/(k-r)$	$\rho = n/(k-r)$
CAONT-RS [Li et al. 2014a]	–	–	–	$\rho = n/k$	$\rho = n/k$
CDStore [Li et al. 2015b]	–	–	–	$\rho = n/k$	$\rho = n/k$
Li et al. [2015]	–	–	–	$\rho = n$	$\rho = n$

ρ, storage blowup factor.

Table VIII. Security Comparison for Deterministic Information Dispersal Schemes

Schemes	Confidentiality	Availability	Side-Channel Resistance	Ownership Authenticity	Data Integrity
CRSSS [Li et al. 2014a]	◯	◯	–	–	×
CAONT-RS [Li et al. 2014a]	◯	◯	–	–	×
CDStore [Li et al. 2015b]	◯	◯	–	–	×
Li et al. [2015]	◯	◯	–	–	×

resulting shares after dispersal to the size of original secret, is n. Compared to CRSSS and CAONT-RS, in which the storage blowup is $n/(k-r)$ and n/k, respectively, Li et al.'s algorithm has a disadvantage in terms of storage efficiency, as its storage blowup is higher than that of the other algorithms.

5.4.2. Comparative Analysis.

Efficiency. Each scheme has been evaluated in terms of storage cost on the server side as well as communication cost over the network. Table VII shows the comparison of the deterministic information dispersal schemes. ρ is the storage blowup, which is the ratio of the size of dispersed shares to the size of original secret. Only the storage blowup factor is considered in the efficiency analysis, as it is the main performance concern for deterministic information dispersal schemes.

Security. Table VIII shows the security comparison between the deterministic information dispersal schemes. Because ensuring access to outsourced data even during system failure is the main concern of these schemes, they satisfy availability. Furthermore, secret sharing–based IDAs are used in these schemes, so they are information-theoretically secure against data breach attacks unless a subset of CSPs is compromised; hence, they also provide data confidentiality. Deterministic information dispersal schemes only consider server-side deduplication, and thus side-channel resistance and ownership authenticity are not applicable in these schemes.

6. OPEN SECURITY ISSUES AND CHALLENGES

Although extensive research has solved many problems regarding the security, privacy, and reliability in a cloud deduplication system, there still remain interesting and pivotal open research challenges. To stimulate research interest and effort in enhancing secure data deduplication, we present several unresolved issues in this section.

6.1. Data Ownership Management

In the case of ownership management and revocation, suppose that multiple users have ownership of a ciphertext outsourced to cloud storage. As time elapses, some users may request the CSP to delete or modify their data, and the CSP then deletes the ownership information of the users from the ownership list for the corresponding data. The revoked users should then be prevented from accessing the data stored in the cloud storage after the deletion or modification request. On the other hand, when a user uploads data that already exists in the cloud storage, the user should be blocked from

accessing the data that was stored before he or she obtained ownership by uploading it.[1] These dynamic ownership changes may occur very frequently in a practical cloud system, and thus it should be carefully managed to avoid the security degradation of the cloud service. However, the previous deduplication schemes have not been able to achieve secure access control in an environment with a dynamic ownership change despite its importance to secure deduplication, because the encryption key in most existing secure deduplication schemes is derived deterministically and rarely updated after the initial key derivation. Therefore, as long as revoked users keep the encryption key, they can access the corresponding data in the cloud storage at any time, regardless of the validity of their ownership. Thus, data ownership management and revocation is one challenging open issue for secure deduplication.

6.2. Achieving Semantically Secure Deduplication

Semantically secure encryption schemes enable clients who care about privacy to encrypt their data on the client side. However, the naive application of semantic encryption may make deduplication impossible, as completely independent ciphertexts are very likely to be produced from identical files. Therefore, reconciling deduplication and semantically secure encryption is a challenging research topic. CE [Douceur et al. 2002] is an effective solution, but it is susceptible to offline brute-force attacks on the hash of the data, especially when the data is predictable, because the encryption is deterministic (i.e., not semantically secure). One effective solution to achieve semantically secure deduplication is to allow an initial uploader to encrypt his data with a randomly generated encryption key and distribute it only to users who share the same data securely [Bellare et al. 2013a; Xu et al. 2013; Bellare et al. 2013b; Duan 2014; Liu et al. 2015]. Unfortunately, some of the schemes require the aid of one or more additional outside KSs, which is a strong assumption that is difficult to meet in practical cloud systems [Bellare et al. 2013b; Duan 2014], and are even susceptible to online brute-force attacks. The other schemes achieve semantic security without the help of additional KSs but enable subsequent uploaders to learn the identities of previous uploaders of the same data [Bellare et al. 2013a; Xu et al. 2013], which violates user privacy, or require every previous uploader to be online [Liu et al. 2015], which incurs significant reductions in both efficiency and deduplication effectiveness. Therefore, achieving semantically secure deduplication without the degradation of deduplication effectiveness or the aid of additional KSs is a pivotal open issue for secure deduplication.

6.3. PoW in Decentralized Deduplication Systems

Outsourced data after deduplication is likely to be vulnerable to data loss or corruption because data deduplication techniques keep only one copy of the data instead of storing multiple copies. To address the reliability of outsourced data, a decentralized approach has been recent focus of research. Decentralized deduplication schemes are applied to multicloud storage architecture, which combines multiple cloud storage services into a single storage service. Many researchers [Li et al. 2014a, 2015, 2015b] have studied ways to enable deduplication of encrypted data outsourced to multicloud storage. Their work has mainly focused on devising deterministic IDAs that transform data into multiple deterministic shares with some redundancy and disperse the shares across multiple CSPs. These approaches protect outsourced data against data corruption to a certain degree and maintain security against inside adversaries, including the CSP. However, they cannot be directly applied to client-side deduplication systems in a multicloud architecture. A PoW solution is required for client-side deduplication to

[1]A multimedia streaming service based on a pay-as-you-go policy in the cloud is a good example of services that have these backward and forward security requirements.

prevent outside adversaries mounting an information leakage attack or unauthorized data access by leveraging deduplication. The security of PoW is generally dependent on the min-entropy of the outsourced data in a view of outside adversaries. Once data with min-entropy k is dispersed into n shares, each share has min-entropy k_i such that $k_i \leq k$ and $\sum_{i=1}^{i=n} k_i = k$. For adversaries, breaking the PoW for each share one by one, which is a divide-and-conquer strategy, is much easier than breaking it for the entire original data at once. Hence, the desired level of PoW security for the outsourced data cannot be guaranteed in a multicloud architecture. We need to further address this problem to achieve efficiency gains using decentralized client-side deduplication systems while also guaranteeing security and availability.

6.4. New Security Threats on Deduplication

We introduce new kinds of security threats that can be exposed to cloud storage systems when deduplication techniques are applied. First, let us consider a scenario where deduplication can be flexibly enabled or disabled on demand. In such flexible deduplication, a cloud storage system may be vulnerable to information leakage attack by adversaries. For instance, suppose that given a set of files F in the storage, deduplication is enabled at time t1 and t3 and disabled at time t2, where t1<t2<t3. By exploiting side channels from deduplication, which is described in Section 3.2.1., an adversary trying to infer whether a certain file f is in F or not may obtain further sensitive information about the time when f is included in F (t1 or t3) in addition to the information of the simple existence of the file.

For the second security threat, let us consider a cloud storage system that supports concurrent read and write block operations. As noted in Paulo and Pereira [2014], applying deduplication to the storage where both read and write operations are frequent causes profound performance degradation to the underlying storage system. Then, by exploiting the performance overhead of deduplication, an adversary would be able to mount a DoS attack and easily incur huge access delay on such cloud storage systems. These new kinds of attack have not been considered yet in the literature, and thus we believe that these problems are also important issues for secure deduplication as promising future work.

7. THE (IM)PRACTICALITY OF SECURE DEDUPLICATION SCHEMES

Security threats to data deduplication are the most critical impediment to the achievement of practical cloud systems, such as information leakage and data corruptions on the cloud storage. As reviewed throughout the previous sections, a variety of state-of-the-art secure deduplication techniques promisingly strengthen the security of cloud storage for each different threat. Despite the successful countermeasures for security issues, however, there is another problem to be overcome before these security solutions are broadly employed in practical applications. We discuss several practicality issues of secure deduplication schemes in this section.

To grasp the current usage of secure deduplication techniques in practical fields, we extensively examined the following major cloud storage services: Amazon [2016a], Dropbox [2016a], Google Drive [2016], OneDrive [2016], Bitcasa [2016], and Wuala [2015]. In addition to storage services, we also examined the following popular storage and deduplication products: OpenDedup [2016], Q-Cloud Protect [Quantum 2016], CloudBerry [2016], DupLESS [Bellare et al. 2013c], CDStore [Li et al. 2015c], Duplicati [2016], and StorReduce [2016]. Specifically, we investigated technical details about security features underlying these services and products by surveying publicly available white papers and other technical documents provided by service providers and product vendors.

Table IX. Current Usages of Secure Deduplication Schemes

		Deduplication				Security			
		Granularity	Location	Boundary	Architecture	Message-Dependent Encryption	PoW	Traffic Obfuscation	Information Dispersal
Service	Amazon S3 [Amazon 2016a]	B	S	Inter	Single	–	–	–	–
	Dropbox [2016a]	B	C/S	Intra	Single	–	–	–	–
	Google Drive [2016]	B	S	Inter	Single	–	–	–	–
	OneDrive [2016]	B	S	Inter	Single	–	–	–	–
	Bitcasa [2016]	F/B	C/S	Inter	Single	MLE	–	–	–
	Wuala [2015] *	F/B	S	Inter	Single	MLE	–	–	–
Product/ Solution	OpenDedup [2016]	F/B	S	Inter	Single	–	–	–	–
	Q-Cloud [Quantum 2016]	F/B	S	Intra	Single	–	–	–	–
	CloudBerry [2016]	B	S	Inter	Single	–	–	–	–
	DupLESS [Bellare et al. 2013c]	F/B	C/S	Inter	Single	DupLESS	–	–	–
	CDStore [Li et al. 2015c]	F	S	Inter	Multi	–	–	–	CDStore
	Duplicati [2016]	B	S	Inter	Single	–	–	–	–
	StorReduce [2016]	B	S	Inter	Single	–	–	–	–

B, block level; F, file level; C, client side; S, server side; Inter, interuser; Intra, intrauser, Single, single cloud; Multi, multicloud.
*Wuala was closed down in 2015.

Table IX presents our survey and analysis results on the usage of secure deduplication techniques in diverse cloud storage services and products. Security features of each service and product were evaluated with respect to our taxonomy of techniques and the evaluation criteria presented in Section 4. Deduplication features were evaluated with regard to the following criteria: granularity (block level vs. file level), location (server side vs. client side), boundary (interuser vs. intrauser), and architecture (single cloud vs. multicloud). The result indicates that despite wide utilization of deduplication systems, relevant security features are rarely deployed in cloud storage services and products. Although certain services allegedly apply security techniques on deduplicated data [Amazon 2016b; Dropbox 2016b; OpenDedup 2016; StorReduce 2016; Quantum 2016], they fail to comply with our security requirements (defined in Section 4.2). For instance, Amazon S3 offers server-side encryption to achieve both goals of deduplication and encryption simultaneously [Amazon 2016b]. It simply allows a storage server to encrypt outsourced data using a conventional encryption algorithm (e.g., AES-256) under secret keys chosen by the server itself. Thus, this approach is totally insecure since the server (i.e., inside adversary) still has access to the content of outsourced data. According to our survey, only two commercial storage services—Bitcasa [2016] and Wuala [2015]—employ message-dependent encryption schemes to enable deduplication over encrypted data. Furthermore, with regard to deduplication products, most of them except DupLESS [Bellare et al. 2013c] and CDStore [Li et al. 2015c] are found to apply none of secure deduplication schemes.

Despite the crucial need for secure deduplication, such low utilization of security techniques in practical applications points out that a large obstacle still remains between the necessity and the effectiveness of current security implementations. In this regard, we observed that there are two reasons for the obstacle: (1) the large gap between theoretical security and practical efficiency required in the real world and (2) the lack of a sufficient number of well-implemented software implementations for secure deduplication. We present more details about these issues in the following sections.

7.1. Large Gap Between Theory and Practice

Secure deduplication schemes generally come with the cost of losing some efficiency in underlying storage systems. Therefore, for many scenarios in the real world, the trade-off between practical efficiency and security should be carefully considered when

deciding on the level of security. A variety of techniques presented in Section 5 achieve diverse security goals under different threat and cloud architecture models. However, some of the techniques require huge computing or network resources, which may eventually make CSPs and product vendors reluctant to adopt secure deduplication schemes in their systems.

For instance, message-dependent encryption (see Section 5.1) is an essential algorithm for cloud storage to enable deduplication over encrypted data. The scheme of Liu et al. [2015] is one of the promising message-dependent encryption schemes because it offers flexibility on underlying cloud storage architecture while simultaneously providing strong data confidentiality even to predictable message distributions. The algorithm of Liu et al.'s scheme is realized by using homomorphic encryption [Paillier 1999; Elgamal 1985], which is a state-of-the-art cryptographic algorithm that allows arithmetic operations on ciphertexts. Despite its novelty, however, the current level of implementations of homomorphic encryption is still far from practical use. Recent benchmarks [Jost et al. 2015] for the fastest implementation of the Paillier homomorphic encryption primitive [Paillier 1999] on an Intel i7-4600U CPU records 0.5msec per encryption, which is 190,000 times slower than that of AES on an Intel i7-980X using AES-NI [Intel 2010]. The computational inefficiency of homomorphic encryption is one of the biggest hurdles for Liu et al.'s scheme to be utilized in practical cloud storage systems.

Another example can be found in PoW schemes (see Section 5.2) and traffic obfuscation schemes (see Section 5.3). To prevent unauthorized data access and information leakage from client-side deduplication, these solutions inevitably generate a certain amount of network traffic at the side of cloud service provider. Although theoretical soundness of security is preserved by these approaches, consuming considerable bandwidth resources is never inexpensive to the service provider. Most providers, such as Amazon Web Services (AWS) and CloudFlare, purchase bandwidth from several Internet service providers (ISPs) to ensure global connectivity to their cloud storage. For example, CloudFlare pays Tier 1 ISPs about $35 for the maximum amount of bandwidth in megabits per second per month [CloudFlare 2014]. Therefore, the cost for applying PoW and traffic obfuscation to cloud storage services would be nonnegligible in practice, which may limit their adoption by the service providers.

7.2. Lack of a Sufficient Number of Software Implementations

Cloud storage systems are constructed on a variety of relevant technologies, including virtualization, storage, network, and security, most of which consist of plenty of complex software implementations. Owing to advances in software engineering, developers in cloud service providers and product vendors do not have to reinvent the wheel to build their storage services and products. Instead, they are likely to integrate software packages or libraries for each technology into the system by purchasing them or obtaining the source codes from open source projects. Hence, utilization of existing software implementations including commercial proprietary packages and open sources are necessary for specific technologies to be productively realized in practice.

When it comes to security technologies, it is critically important to use the correctness and security-proved software instead of reimplementing it from scratch without profound knowledge of complex cryptography and security. Implementation without them may lead to serious vulnerabilities to underlying software systems. For instance, it was reported that a substantial rate of security incidents from Android apps is caused by poor implementations of security solutions such as SSL/TLS [Fahl et al. 2012].

In this context, we point out that another obstacle limiting the prevalent use of secure deduplication schemes originates in a lack of well-implemented software for the technology. Figure 2 shows the number of software implementations for secure

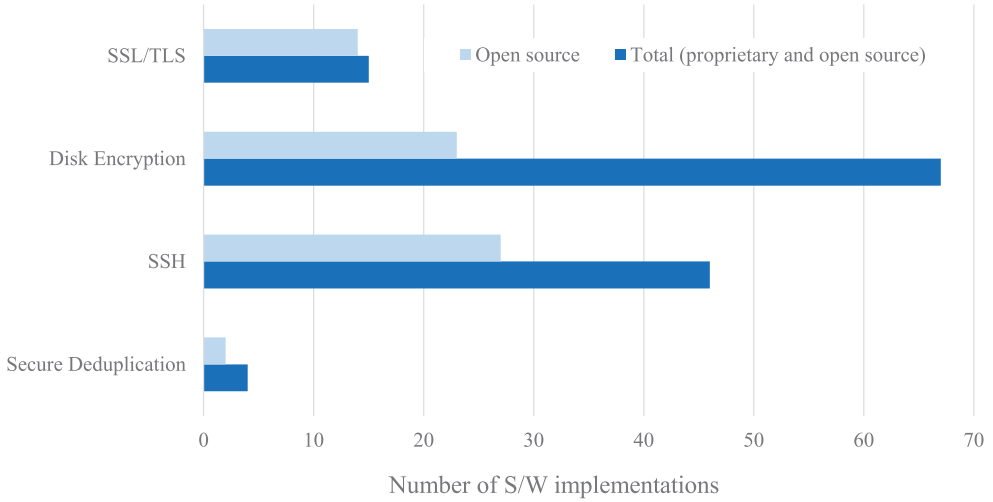

Fig. 2. Number of existing software implementations for each security solution (in 2016).

deduplication schemes in Table IX, which is denoted by "Secure Deduplication." For comparison purposes, we also investigated those of Secure Shell (SSH) [Fingolfin 2016], disk encryption [Wikipedia 2016a], and SSL/TLS [Wikipedia 2016b] in Figure 2. According to the analysis, we found that there are only four implementations of secure deduplication techniques in total (using proprietary software packages or open sources) at present. Among them, two open-source implementations—DupLESS [Bellare et al. 2013c] and CDStore [Li et al. 2015c]—have been made public for academic purposes rather than for practical use. Compared to SSH, disk encryption, and SSL/TLS, which have a higher rate of practical usage with plenty of software implementations, there is much less software available for secure deduplication. For the successful adoption of secure deduplication to practical applications, it is crucial to develop correctness and security-proved software for the primitive secure deduplication techniques and provide it to cloud service developers.

8. CONCLUSION

Secure deduplication is an effective technique for minimizing cloud storage space while preserving the security and privacy of cloud data. In this article, we introduced a taxonomy of state-of-the-art deduplication techniques according to the deduplication system environment and security goals considering a diverse range of threats. Specifically, we surveyed existing secure deduplication schemes and classified them into cryptographic solutions and protocol solutions, including message-dependent encryption, PoW, traffic obfuscation, and deterministic information dispersal approaches. We then rigorously analyzed and compared them in terms of efficiency and security, and provided the advantages and disadvantages of each scheme. Finally, we discussed future research directions for secure deduplication in cloud storage. To the best of our knowledge, this is the first survey and discussion of secure deduplication techniques. Although deduplication systems and their benefits are now widely accepted by most current cloud service providers, there are many practical concerns regarding the security and privacy of cloud data. Because there are still interesting and pivotal open research challenges, we predict that research on secure deduplication will continue to grow in the forthcoming years.

REFERENCES

Martín Abadi, Dan Boneh, Ilya Mironov, Ananth Raghunathan, and Gil Segev. 2013. Message-locked encryption for lock-dependent messages. In *Advances in Cryptology—CRYPTO 2013*. Lecture Notes in Computer Science, Vol. 8042. Springer, 374–391.

Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. 2010. RACS: A case for cloud storage diversity. In *Proceedings of the 1st ACM Symposium on Cloud Computing*. 229–240.

Amazon. 2016a. Amazon S3. Retrieved December 7, 2016, from https://aws.amazon.com/s3.

Amazon. 2016b. Protecting Data Using Server-Side Encryption. Retrieved December 7, 2016, from http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html.

Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. 2007. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*. 598–609. DOI:http://dx.doi.org/10.1145/1315245.1315318

Mihir Bellare and Sriram Keelveedhi. 2015. Interactive message-locked encryption and secure deduplication. In *Advances in Cryptology—EUROCRYPT 2013*. Lecture Notes in Computer Science, Vol. 9020. Springer, 516–538.

Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. 2013a. Message-locked encryption and secure deduplication. In *Advances in Cryptology—EUROCRYPT 2013*. Lecture Notes in Computer Science, Vol. 7881. Springer, 296–312.

Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. 2013b. DupLESS: Server-aided encryption for deduplicated storage. In *Proceedings of USENIX Security Symposium 2013*. 179–194.

Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. 2013c. Source code of DupLESS. Retrieved December 7, 2016, from http://cseweb.ucsd.edu/~skeelvee/dupless/.

Bitcasa. 2016. Home Page. Retrieved December 7, 2016, from https://www.bitcasa.com.

G. R. Blakley and C. Meadows. 1985. Security of ramp schemes. In *Advances in Cryptology—CRYPTO 1984*. Lecture Notes in Computer Science, Vol. 196. 242–268. DOI:http://dx.doi.org/10.1007/3-540-39568-7_20

J. Blasco, R. Di Pietro, A. Orfila, and A. Sorniotti. 2014. A tunable proof of ownership scheme for deduplication using Bloom filters. In *Proceedings of the 2014 IEEE Conference on Communications and Network Security (CNS'14)*. 481–489.

Deepak R. Bobbarjung, Suresh Jagannathan, and Cezary Dubnicki. 2006. Improving duplicate elimination in storage systems. *ACM Transactions on Storage* 2, 4, 424–448.

Andrei Z. Broder. 2000. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching*. Lecture Notes in Computer Science, Vol. 1848. Springer, 1–10. DOI:http://dx.doi.org/10.1007/3-540-45123-4_1

Cisco. 2015. CISCO Global Cloud Index (2012-2017). Available at http://www.cisco.com/c/en/us/solutions/service-provider/global-cloud-index-gci/index.html.

CloudBerry. 2016. Home Page. Retrieved December 7, 2016, from http://www.cloudberrylab.com.

CloudFlare. 2014. The Relative Cost of Bandwidth Around the World. Retrieved December 7, 2016, from https://blog.cloudflare.com/the-relative-cost-of-bandwidth-around-the-world/.

Roberto Di Pietro and Alessandro Sorniotti. 2012. Boosting efficiency and security in proof of ownership for deduplication. In *Proceedings of the 7th ACM Symposium on Information, Computer, and Communications Security (ASIACCS'12)*. 81–91.

John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, and Marvin Theimer. 2002. Reclaiming space from duplicate files in a serverless distributed file system. In *Proceedings of the 2002 22nd International Conference on Distributed Computing Systems*. IEEE, Los Alamitos, CA, 617–624.

Dropbox. 2016a. Home Page. Retrieved December 7, 2016, from http://www.dropbox.com.

Dropbox. 2016b. *Dropbox Business Security: A Dropbox Whitepaper*. Retrieved December 7, 2016, from https://www.dropbox.com/static/business/resources/Security_Whitepaper.pdf.

Yitao Duan. 2014. Distributed key generation for encrypted deduplication. In *Proceedings of the 6th Edition of the ACM Cloud Computing Security Workshop (CCSW'14)*. 57–68.

Duplicati. 2016. Home Page. Retrieved December 7, 2016, from http://www.duplicati.com.

Cynthia Dwork. 2008. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*. Lecture Notes in Computer Science, Vol. 4978. Springer, 1–19. DOI:http://dx.doi.org/10.1007/978-3-540-79228-4_1

Stefan Dziembowski. 2006. Intrusion-resilience via the bounded-storage model. In *Theory of Cryptography—TCC 2006*. Lecture Notes in Computer Science, Vol. 3876. Springer, 207–224. DOI:http://dx.doi.org/10.1007/11681878_11

Taher Elgamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 4, 469–472. DOI:http://dx.doi.org/10.1109/TIT.1985.1057074

Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, Lars Baumgärtner, and Bernd Freisleben. 2012. Why Eve and Mallory love Android: An analysis of Android SSL (In)Security. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)*. 50–61.

Fingolfin. 2016. SSH Implementation Comparison. Retrieved December 7, 2016, from http://ssh-comparison.quendi.de/.

Min Fu, Dan Feng, Yu Hua, Xubin He, Zuoning Chen, Wen Xia, Yucheng Zhang, and Yujuan Tan. 2015. Design tradeoffs for data deduplication performance in backup workloads. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST'15)*. 331–344.

Lorena González-Manzano and Agustín Orfila. 2015. An efficient confidentiality-preserving proof of ownership for deduplication. *Journal of Network and Computer Applications* 50, 49–59.

Google Drive. 2016. Home Page. Retrieved December 7, 2016, from https://www.google.com/drive/

Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. 2011. Proofs of ownership in remote storage systems. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. 491–500.

Danny Harnik, Oded Margalit, Dalit Naor, Dmitry Sotnikov, and Gil Vernik. 2012. Estimation of deduplication ratios in large data sets. In *Proceedings of the 2012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST'12)*. 1–11.

Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. 2010. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security and Privacy Magazine* 8, 6, 40–47.

O. Heen, C. Neumann, L. Montalvo, and S. Defrance. 2012. Improving the resistance to side-channel attacks on cloud storage services. In *Proceedings of the 2012 5th International Conference on New Technologies, Mobility, and Security (NTMS'12)*. 1–5.

Intel. 2010. *Breakthrough AES Performance with Intel AES New Instructions*. White Paper. Retrieved December 7, 2016, from http://www.intel.it/content/dam/www/public/us/en/documents/white-papers/aes-breakthrough-performance-paper.pdf.

Keren Jin and Ethan L. Miller. 2009. The effectiveness of deduplication on virtual machine disk images. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*. 7.

Christine Jost, Ha Lam, Alexander Maximov, and Ben J. M. Smeets. 2015. Encryption Performance Improvements of the Paillier Cryptosystem. Retrieved December 7, 2016, from https://eprint.iacr.org/2015/864

Ari Juels and Burton S. Kaliski Jr. 2007. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*. 584–597. DOI:http://dx.doi.org/10.1145/1315245.1315317

Nesrine Kaaniche and Maryline Laurent. 2014. A secure client side deduplication scheme in cloud storage environments. In *Proceedings of the 2014 6th International Conference on New Technologies, Mobility, and Security (NTMS'14)*. 1–7. DOI:http://dx.doi.org/10.1109/NTMS.2014.6814002

Michal Kaczmarczyk, Marcin Barczynski, Wojciech Kilian, and Cezary Dubnicki. 2012. Reducing impact of data fragmentation caused by in-line deduplication. In *Proceedings of the 5th Annual International Systems and Storage Conference*. 15.

Dongyoung Koo, Junbeom Hur, and Hyunsoo Yoon. 2014. Secure and efficient deduplication over encrypted data with dynamic updates in cloud storage. In *Proceedings of the International Symposium on Frontier and Innovation in Future Computing and Communications (FCC'14)*. 229–235. DOI:http://dx.doi.org/10.1007/978-94-017-8798-7_28

Seungkwang Lee and Dooho Choi. 2012. Privacy-preserving cross-user source-based data deduplication in cloud storage. In *Proceedings of the 2012 International Conference on ICT Convergence (ICTC'12)*. 329–330.

J. Li, X. Chen, X. Huang, S. Tang, Y. Xiang, M. Hassan, and A. Alelaiwi. 2015. Secure distributed deduplication systems with improved reliability. *IEEE Transactions on Computers* 64, 12, 3569–3579. DOI:http://dx.doi.org/0.1109/TC.2015.2401017

J. Li, X. Chen, M. Li, P. P. C. Lee, and W. Lou. 2014a. Secure deduplication with efficient and reliable convergent key management. *IEEE Transactions on Parallel and Distributed Systems* 25, 6, 1615–1625.

J. Li, Y. K. Li, X. Chen, P. P. C. Lee, and W. Lou. 2015a. A hybrid cloud approach for secure authorized deduplication. *IEEE Transactions on Parallel and Distributed Systems* 26, 5, 1206–1216.

Mingqiang Li, Chuan Qin, and Patrick P. C. Lee. 2015b. CDStore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. In *Proceedings of the 2015 USENIX Annual Technical Conference (ATC'15)*. 1–20.

Mingqiang Li, Chuan Qin, and Patrick P. C. Lee. 2015c. Source Code of CDStore. Retrieved December 7, 2016, from https://github.com/chintran27/CDStore

Mingqiang Li, Chuan Qin, Patrick P. C. Lee, and Jin Li. 2014b. Convergent dispersal: Toward storage-efficient security in a coud-of-clouds. In *Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'14)*.

Chih Wei Ling and Anwitaman Datta. 2014. InterCloud RAIDer: A do-it-yourself multi-cloud private data backup system. In *Distributed Computing and Networking*. Lecture Notes in Computer Science, Vol. 8314. Springer, 453–468. DOI:http://dx.doi.org/10.1007/978-3-642-45249-9_30

Chuanyi Liu, Yancheng Wang, and Jie Lin. 2014. A policy-based de-duplication mechanism for encrypted cloud storage. *Journal of Computational Information Systems* 10, 6, 2297–2304.

J. Liu, N. Asokan, and B. Pinkas. 2015. Secure deduplication of encrypted data without additional independent servers. In *Proceedings of the 22th ACM Conference on Computer and Communications Security (CCS'15)*. 874–885.

Nagapramod Mandagere, Pin Zhou, Mark A. Smith, and Sandeep Uttamchandani. 2008. Demystifying data deduplication. In *Proceedings of the ACM/IFIP/USENIX Middleware 2008 Conference Companion (Companion'08)*. 12–17.

Ralph C. Merkle. 1990. A certified digital signature. In *Advances in Cryptology—CRYPTO 1989*. Lecture Notes in Computer Science, Vol. 435. Springer, 218–238. DOI:http://dx.doi.org/10.1007/0-387-34805-0_21

Dutch T. Meyer and William J. Bolosky. 2012. A study of practical deduplication. *ACM Transactions on Storage* 7, 4, 1–20.

Meixia Miao, Jianfeng Wang, Hui Li, and Xiaofeng Chen. 2015. Secure multi-server-aided data deduplication in cloud computing. *Pervasive and Mobile Computing* 24, C, 129–137. DOI:http://dx.doi.org/10.1016/j.pmcj.2015.03.002

Martin Mulazzani, Sebastian Schrittwieser, Manuel Leithner, Markus Huber, and Edgar R. Weippl. 2011. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In *Proceedings of the 2011 USENIX Security Symposium*.

Wee Keong Ng, Yonggang Wen, and Huafei Zhu. 2012. Private data deduplication protocols in cloud storage. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC'12)*. 441–446.

OneDrive. 2016. Home Page. Retrieved December 7, 2016, from https://onedrive.live.com

OpenDedup. 2016. Home Page. Retrieved December 7, 2016, from http://www.opendedup.org

Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT 1999*. Lecture Notes in Computer Science, Vol. 1592. Springer, 223–238. DOI:http://dx.doi.org/10.1007/3-540-48910-X_16

João Paulo and José Pereira. 2014. A survey and classification of storage deduplication systems. *ACM Computing Surveys* 47, 1, 1–30. DOI:http://dx.doi.org/10.1145/2611778

Torben Pryds Pedersen. 1992. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO 1992*. Lecture Notes in Computer Science, Vol. 576. Springer, 129–140. DOI:http://dx.doi.org/10.1007/3-540-46766-1_9

Tobias Pulls. 2012. (More) side channels in cloud storage. *Privacy and Identity Management for Life* 375, 102–115.

Pasquale Puzio, Refik Molva, Melek Onen, and Sergio Loureiro. 2013. ClouDedup: Secure deduplication with encrypted data for cloud storage. In *Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom'13)*. 363–370.

Quantum. 2016. Home Page. Retrieved December 7, 2016, from http://www.quantum.com/products/cloudservices/q-cloud/index.aspx.

Michael Rabin. 1981. *Fingerprinting by Random Polynomials*. Center for Research in Computing Technology, Aiken Computation Laboratory, Harvard University, Cambridge, MA.

Michael Rabin. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM* 36, 335–348.

Jason K. Resch, James S. Plank, Jason K. Resch, and James S. Plank. 2011. AONT-RS: Blending security and performance in dispersed storage systems. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11)*.

Adi Shamir. 1979. How to share a secret. *Communications of the ACM* 22, 11, 612–613. DOI:http://dx.doi.org/10.1145/359168.359176

Youngjoo Shin, Junbeom Hur, and Kwangjo Kim. 2012. Security Weakness in the Proof of Storage with Deduplication. Retrieved December 7, 2016, from https://eprint.iacr.org/2012/554

Youngjoo Shin and Kwangjo Kim. 2015. Differentially private client-side data deduplication protocol for cloud storage services. *Security and Communication Networks* 8, 2114–2123. DOI:http://dx.doi.org/10.1002/sec.1159

Jan Stanek, Alessandro Sorniotti, Elli Androulaki, and Lukas Kencl. 2013. A secure data deduplication scheme for cloud storage. In *Financial Cryptography and Data Security*. Lecture Notes in Computer Science, Vol. 8437. Springer, 99–118.

Mark W. Storer, Kevin Greenan, Darrell D. E. Long, and Ethan L. Miller. 2008. Secure data deduplication. In *Proceedings of the 4th ACM International Workshop on Storage Security and Survivability (StorageSS'08)*. 1–10.

StorReduce. 2016. Home Page. Retrieved December 7, 2016, from http://www.storreduce.com

Mi Wen, Kejie Lu, Jingsheng Lei, Fengyong Li, and Jing Li. 2015. BDO-SD: An efficient scheme for big data outsourcing with secure deduplication. In *Proceedings of the 2015 IEEE INFOCOM Workshop on Big Data Security*. 1–6.

Wikipedia. 2016a. Comparison of Disk Encryption Software. Retrieved December 7, 2016, from https://en.wikipedia.org/wiki/Comparison_of_disk_encryption_software.

Wikipedia. 2016b. Comparison of TLS Implementations. Retrieved December 7, 2016, from https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations.

Wuala. 2015. Home Page. Retrieved December 7, 2016, from https://en.wikipedia.org/wiki/Wuala.

Jia Xu, Ee-Chien Chang, and Jianying' Zhou. 2011. Leakage-Resilient Client-Side Deduplication of Encrypted Data in Cloud Storage. Retrieved December 7, 2016, from https://eprint.iacr.org/2011/538

Jia Xu, Ee-Chien Chang, and Jianying Zhou. 2013. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In *Proceedings of the 8th ACM Symposium on Information, Computer, and Communications Security (ASIACCS'13)*. 195.

Chao Yang, Jianfeng Ma, and Jian Ren. 2015. Provable ownership of encrypted files in de-duplication cloud storage. *Ad Hoc and Sensor Wireless Networks* 26, 43–72.

Chao Yang, Jian Ren, and Jianfeng Ma. 2013a. Provable ownership of file in de-duplication cloud storage. In *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM'13)*. 695–700.

Chao Yang, Jian Ren, and Jianfeng Ma. 2013b. Provable ownership of files in deduplication cloud storage. *Security and Communication Networks* 8, 14, 2457–2468. DOI: http://dx.doi.org/10.1002/sec.784

Jiawei Yuan and Shucheng Yu. 2013. Secure and constant cost public cloud storage auditing with deduplication. In *Proceedings of the 2013 IEEE Conference on Communications and Network Security (CNS'13)*. 145–153.

Qingji Zheng and Shouhuai Xu. 2012. Secure and efficient proof of storage with deduplication. In *Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY'12)*. 1–12.

Yifeng Zheng, Xingliang Yuan, Xinyu Wang, Jinghua Jiang, Cong Wang, and Xiaolin Gui. 2015. Enabling encrypted cloud media center with secure deduplication. In *Proceedings of the 10th ACM Symposium on Information, Computer, and Communications Security (ASIACCS'15)*. 63–72.

Y. Zhou, D. Feng, W. Xia, M. Fu, F. Huang, Y. Zhang, and C. Li. 2015. SecDep: A user-aware efficient fine-grained secure deduplication scheme with multi-level key management. In *Proceedings of the 31st International Conference on Massive Storage Systems and Technology (MSST'15)*.

Received February 2016; revised October 2016; accepted November 2016