On Cloud Storage and the Cloud of Clouds Approach

Daniel Slamanig, Christian Hanser
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology (TUG)
Inffeldgasse 16a, 8010 Graz, Austria
{daniel.slamanig|christian.hanser}@iaik.tugraz.at

***Abstract*—Many recently proposed cloud storage architectures build a single virtual cloud storage system by using a combination of diverse commercial cloud storage services - the so called *cloud of clouds approach*. Thereby, the data to be stored is dispersed among different (independent) cloud storage providers in a redundant way. This is commonly accomplished either by naively replicating the data to several providers (storing an entire copy of a file at each provider) or by dispersing suitably encoded data, i.e., only a certain threshold of file fragments is required for reconstruction of a file. Furthermore, since many vendors of commercial cloud storage services do not provide adequate means to securing the cloud from within the cloud infrastructure, many recently proposed cloud storage architectures (transparently) add relevant security and privacy features from the outside. In doing so, they are mainly trying not to affect the cloud provider's interfaces and inner workings.**

In this paper we take a closer look at distributed cloud storage systems. We provide an overview of information dispersal strategies to realise reliable distributed cloud storage systems and provide an overview of state-of-the-art cloud storage approaches. Then, we analyse them with respect to security properties. Furthermore, we discuss the lack of privacy features and in particular features to provide access privacy in existing distributed cloud storage systems, which is an important direction for future research on distributed cloud storage.

***Index Terms*—Distributed cloud storage, information dispersal, state of the art, storage security, security, access privacy.**

I. Introduction

Cloud storage is an increasingly popular class of online services for archiving, backup, sharing of data, synchronisation of multiple devices and it is also envisioned for future primary storage of (enterprise) data. Meanwhile there are lots of services from infrastructure providers, like Amazon S3, Microsoft Windows Azure Blob Storage, Nirvanix SDN, Rackspace, etc. which provide simple (web based) interfaces to their block-oriented storage services. These services are focusing on enterprises and may be used as storage backend for arbitrary applications in order to improve availability due to time and location independent access to enterprise data. This seems absolutely realistic as long as client-side caching and weak consistency models are used to avoid accessing the cloud on every operation. Such services often also serve as storage infrastructure for popular end customer oriented cloud storage services like Dropbox, Mozy, TeamDrive, Wuala, etc., which allow easy data outsourcing and sharing with other users.

Despite the obvious advantages of cloud storage being among others ubiquitous access to data, immediate scalability and the pay-per-usage billing model, there are still concerns which hinder a widespread adoption. These concerns are mainly devoted to missing or inadequate security and privacy related features, requiring customers to fully trust in the integrity of the cloud provider as well as the provider's security practices. However, besides securing the cloud from within the cloud infrastructure, an alternative possibility is to (transparently) add relevant security and privacy features from the outside without affecting the cloud provider's interfaces and inner workings. Within the last few years several approaches to eliminate security deficiencies within state of the art cloud storage systems have been proposed [1]–[11]. Thereby focusing on confidentiality, authenticity, integrity, consistency of reads and writes in a multi-user setting as well as (public) auditability or verifiability. We provide an overview of existing approaches in §III and evaluate them with respect to the aforementioned properties in §III-K.

Another important issue considered in the design of various recent approaches to cloud storage is to increase the availability of stored data. This is extremely important when using the cloud as the primary storage site. Recent incidents [12], [13] indicate that, despite the assumed high availability guarantees of the cloud, outages occur in practice and various concepts to mitigate this risk – usually applying the so-called *cloud of clouds* approach – have been proposed. Basically, such approaches build a virtual cloud storage by using a combination of diverse commercial cloud storage services. The improved availability is obtained by means of redundancy, as it is well known from traditional techniques for improving reliability in disk storage and distributed storage systems. This is commonly accomplished either by naively replicating the data to several providers (storing an entire copy of a file at each provider) or by dispersing suitably encoded data, i.e., sharing a file over n providers such that k shares of the file suffice for reconstruction. Such approaches reduce the risk of being influenced from single cloud provider outages and provide (at least to some degree) resistance to loss or corruption of data at cloud providers. Also, it reduces the customer's risk for the so-called "vendor lock-in" problem, i.e., that it can become prohibitively expensive for a client to switch from one provider to another. For instance, the Amazon S3 service

did initially not charge for uploading data, but on June 30 2010 Amazon started charging 15¢ per GB. Such changes in a provider's business model may motivate customers to change their storage provider, which should be possible with a reasonable cost overhead [5].

A. Contribution

Our main contributions of this work are:

- We provide an overview of information dispersal strategies applied in existing distributed cloud storage approaches.
- We provide an overview of proposed architectures for (distributed) cloud storage, analyse them with respect to security related properties and provide a compact overview of supported features.
- We discuss the lack of access privacy features in state-of-the-art distributed cloud storage architectures.

B. Paper Organisation

The rest of this paper is organised as follows. In §II we present the idea of distributed cloud storage, important issues in context of this paper and provide an overview of information dispersal strategies. In §III we provide an overview of state of the art distributed cloud storage systems and evaluate them with respect to several security properties. In §IV we discuss access privacy features for distributed cloud storage systems and finally §V concludes our paper.

II. Distributed Cloud Storage

In distributed cloud storage systems, data is redundantly dispersed among several independent cloud storage providers. As already mentioned, this improves availability of data, reduces the risk of data loss and the risk of a vendor lock-in. Furthermore, since one integrates different providers into one single virtual provider, it seems arguable to assume non-cooperation of a fraction of the single providers, which helps to improve confidentiality by construction. This still leaves the problem of a *mobile adversary* who can corrupt all providers over the system's lifetime but not all within the same time period. However, this can be solved by techniques from proactive cryptography [14] or by a reactive test-and-redistribute technique [2] and does not affect our discussion.

A. Relevant Issues

In general we can distinguish between enterprise solutions and solutions thought for private customers, e.g., home users. Enterprises usually have lots of users working in parallel and in such a scenario it is reasonable to use a proxy, e.g., [6], within the enterprise's network perimeter to coordinate actions by different clients and to let the proxy perform all actions on behalf of the clients. Note that inter-enterprise data sharing usually will require one proxy per enterprise and synchronisation of these proxies as applied in [5]. For private customers one could also rely on proxies, e.g., run them in the cloud – this could be some active trusted component or an untrusted component as it is the case in [4][1]. Alternatively, one can use a fully decentralised approach which makes it more convenient to use with multiple devices like Desktop PCs, Laptops, Smartphones, etc. Approaches which make use of a proxy allow much more straightforward realisations of several security and privacy related features including access control and access privacy. This is due to the fact that the proxy can be seen as a single client accessing the cloud and many otherwise tricky to realise aspects, e.g., confidentiality inclduing key distribution, become very simple to handle, since it can be realised at the proxy outside the cloud.

Private customers are likely to use the cloud storage as a simple backup infrastructure and as designated place to store and share data, while enterprises will more likely use it as a full-fledged virtual file system.

The subsequent issues are particularly relevant:

- I1: Is the data accessed by a single client or by multiple clients (potentially having different privileges).
- I2: If data is shared among multiple clients, is there a single-writer multi-reader (SWMR) setting, or is there a multiple reader and writer (MRMW) setting.
- I3: Is the set of authorised clients dynamic, i.e., can change over time, or is it known and fixed in advance.
- I4: Is the data static or dynamic, i.e., does the data change over time or not. Here, dynamic means i) that files can be added or removed and and ii) that changes to existing files can be made over time.
- I5: Do clients directly interact with the cloud storage providers or is all client communication mediated via a proxy.

B. Information Dispersal

Information dispersal is a popular means to improve availability, reliability and also confidentiality of data in storage systems by decomposing a file into n (smaller) pieces and storing these pieces at n different storage sites. Thereby, the basic idea can be considered as a (k, n)-threshold mechanisms, which means that any k of the n pieces are sufficient to reconstruct the original file. Regarding confidentiality, a common assumption is that no subset of cardinality greater than $k-1$ of the storage sites will cooperate or will be compromised simultaneously such that the content of a file is protected from adversaries. Below, we review the most common techniques for information dispersal used within distributed cloud storage systems, whereas we denote the information to be dispersed by f which is in our case a file or file-block.

1) Shamir's Secret Sharing: In [15] Shamir proposed to represent an information f that should be dispersed as the constant term of an otherwise randomly chosen polynomial $p(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + f$ over some finite field $\mathbb{F}$. Dispersing a file is done by giving the so-called share $p(i)$ to storage site i, $1 \leq i \leq n$. Given any set S of cardinality at least k of these n shares and denote the set of indices corresponding to shares in S by I_S, one can use Lagrange interpolation to compute f as $f = \sum_{j \in I_S} \lambda_j p(j)$, whereas $\lambda_j = \prod_{i \in I_S \setminus \{j\}} j(j-i)^{-1}$ and all computations are in $\mathbb{F}$. The confidentiality guarantees of this threshold scheme are

[1] They, however, do not consider a cloud of cloud approach.

information theoretic, i.e., any $k-1$ shares do not provide any information about f. But disadvantages of this simple method are an increase in storage requirements by a factor n and no inherent error-correction capabilities, i.e., if any of the shares in S is malformed (which cannot be detected in this scheme[2]) then the reconstruction will not yield f.

2) Rabin's Approach: Rabin [17] introduced the notion of information dispersal algorithms (IDAs). He proposed the use of non-systematic erasure codes instead of polynomial secret sharing as proposed by Shamir. Basically, f is viewed as a vector $\vec{f}$ with k elements and one chooses an "invertible"[3] $n \times k$ dispersal matrix $\mathbf{D}$ and the codeword is obtained as $\vec{c} = \mathbf{D} \cdot \vec{f}$. Reconstruction is achieved as $\vec{f} = \mathbf{D'}^{-1} \cdot \vec{c'}$, whereas $\vec{c'}$ contains the k out of n elements of the codeword queried from the storage sites and $\mathbf{D'}$ the corresponding submatrix of $\mathbf{D}$. Using this approach, it is possible to reduce the storage "blowup" to a factor of n/k. Although more efficient in computation and storage requirements, the proposed IDA does not provide information theoretic confidentiality guarantees and in fact provides extremely weak guarantees. If confidentiality is required, Rabin notes that it is recommended to encrypt the data prior to applying the IDA, which requires an additional concept for key distribution.

3) Secret Sharing Made Short: Krawczyk [18] asked the question whether it is possible to make shares in a secret sharing scheme shorter when requiring "only" computational instead of information theoretic security. He proposes to use any IND-CPA secure symmetric encryption scheme in the following (obvious) way: Choose a random secret key sk of a suitable symmetric encryption scheme and encrypt f using sk. To the resulting ciphertext apply an IDA which results in $\vec{c}$ containing n codewords as elements. Then one uses polynomial secret sharing to compute n shares $s_1, \ldots, s_n$ of the key sk. Note that the field $\mathbb{F}$ used for secret sharing can be much smaller as in Shamir's original approach, since the secret is a small symmetric key. Finally, every storage site i receives $(\vec{c}_i, s_i)$ and the reconstruction of f works the obvious way. In addition to the extra storage requirements of IDA, every storage site needs to store one small share of the key. In [19] the authors show that this construction can be proven to be secure if the used encryption scheme also satisfies one-query key-unrecoverability, which is true for practical encryption schemes.

4) AONT-RS: Very recently, the authors of [9] proposed an alternative approach to enrich Rabin's IDA with confidentiality. In contrast to [18] it avoids the use of polynomial secret sharing. The basic idea is to employ a variant of Rivest's All-or-nothing transform (AONT) [20] as a preprocessing step for the data before using an IDA. It works as follows: consider f to be a vector $\vec{f}$ with $k+1$ elements, whereas the $k+1$'th element is a canary – some known fixed value for integrity checking. Then one chooses a random secret key sk of a suitable symmetric encryption scheme (G, E, D) and computes the ciphertext elements as $\vec{e}_i = \vec{f}_i \oplus E(i+1, sk)$. After encrypting all $k+1$ elements one computes $\vec{e}_{k+2} = sk \oplus H(\vec{e}_i || \ldots || \vec{e}_{k+1})$ as the final block, whereas H is a collision-resistant cryptographic hash function. Note that only if all of the first $k+1$ elements of $\vec{e}$ are available, the key sk can be reconstructed and thus f decrypted. The canary element is then used to check the integrity. Since the key is implicitly included into the data, the scheme does not require to share the key. After this preprocessing step a systematic erasure code, i.e., a Reed-Solomon (RS) code, is applied. Consequently, the first $k+2$ rows of $\mathbf{D}$ compose a $k+2 \times k+2$ identity matrix. Then, one computes $\vec{c} = \mathbf{D} \cdot \vec{e}$ and the resulting n elements of the codeword vector are stored at the different storage sites. The systematic code is employed to save computations for encoding and the reconstruction of f works the obvious way.

5) IP-ECC: Integrity-protected error correcting codes (IP-ECC) were introduced with HAIL [2] and are a cryptographic primitive that acts both as a message authentication code (MAC) and an error-correcting (or erasure) code. This recent primitive achieves cross server redundancy through ECC and at the same time represents a corruption resilient MAC on the underlying data. The construction of an IP-ECC in [2] is based on a $(n, l, n-l+1)$ Reed-Solomon (RS) code and can also be be adapted for systematic RS codes (l is the number of elements in the file vector). To tag a file f, it is encoded under the RS code, and then one applies a suitable pseudo-random function (PRF) to the last s code symbols (for $1 \leq s \leq n$ being a system parameter), obtaining a MAC on each of those s code symbols. In their concrete construction they use their UMAC (universal MAC) construction, which is a combination of a keyed PRF and a keyed universal hash function. A codeword is considered valid if at least one of its last s symbols are valid MACs under UMAC on the decoded file f.

To disperse a file to different cloud storage providers, any of the above discussed approaches may be used.

III. Cloud Storage Approaches

Subsequently we review recent approaches to (distributed) cloud storage build on top of existing cloud storage services and enhancing them with additional features. Since they are designed with different goals and applications in mind, e.g., enterprise versus private customers, they provide quite different features. In §III-K we then provide a compact overview of the properties supported by the different approaches.

A. DepSKY

In [7] the author propose a distributed cloud storage system which is capable to integrate different cloud storage services and abstracting away details of the cloud storage providers by offering an object store interface to its clients. For all stored data objects, integrity is verifiable by means of digital signatures contained in the metadata of every object. The authors provide two versions, namely DepSKY-A and DepSKY-CA. The former set of protocols is designed to only improve availability by replicating data on several cloud storage providers and the data is stored in cleartext. The

[2]Detection could be realised by verifiable secret sharing [16].

[3]All $k \times k$ submatrices for all combinations of k rows of $\mathbf{D}$ need to yield invertible matrices.

latter version (DepSKY-CA) provides improved availability and confidentiality of the data. Data is encrypted, then encoded using an erasure code and shares for the encryption key are computed using threshold secret sharing. The key shares along with slices of the encoded file are then stored at different providers (see §II-B3 for a discussion of this approach). They also provide read freshness by integrating version numbers into the metadata of the data objects. Their protocol design is for a single-writer multi-reader setting, but they also propose a lock/lease protocol to enforce consistency in the presence of multiple writers to the same data object. However, the multi-writer setting requires communication among the clients, which seems rather impractical for many applications.

B. HAIL

In [2] the authors propose a concept for distributed cloud storage providing high availability by unifying the use of file redundancy within a cloud provider and across independent cloud providers. HAIL uses so-called integrity-protected error correcting codes (IP-ECC) (see §II-B5) by which it achieves cross server redundancy through ECC and at the same time represents a corruption resilient MAC on the underlying data. They additionally use proofs of retrievability (PoRs) [21] to enable cloud storage providers to prove to a client that all the stored data is still available and reallocate corrupted data when failures are detected (test-and-redistribute technique). In particular, when detecting a fault in one cloud provider's storage, the client recovers the corrupted shares from cross-server redundancy and resets the faulty cloud provider with a correct share. Their distributed PoR allows a set of cloud providers to prove to a client via a challenge-response protocol that a client can recover a file with overwhelming probability. They do not require to store any check values at the client (like within other PoR constructions), but obtain such check values from the collection of cloud providers itself. HAIL requires the cloud to execute code and this should be done in the cloud and not at some client side proxy, since access to all files is required. Otherwise this would introduce too much latency. However, HAIL only supports static files and no efficient file updates are supported.

C. IRIS

In [6] the authors propose an authenticated file system which lets enterprises store data in the cloud and be resilient against potentially untrustworthy cloud providers. It enables an enterprise tenant or an external auditor to verify the integrity and freshness of any data retrieved from the file system and supports typical file system operations. It is designed to support updates from multiple clients in parallel (without blocking) and to handle all existing file system operations (including delete, move and truncate) with minimum overhead and being fully transparent to the clients. It uses an inter-enterprise proxy (the portal) that mediates file system operations between clients and the cloud and caches most recently accessed blocks as well as aggregates recent block updates. The portal also maintains error correcting information for the entire file system and when the portal detects corruption through an auditing protocol, these parities enable recovery of lost or corrupted data. Parities, MACs and Merkle-trees for data and metdata are backed up to the cloud on a regular basis. It is the first known approach that realises dynamic PoRs, i.e., PoRs that support efficient file updates. This is non trivial, since updates partially reveal the code structure, but updates should be efficient and only involve a small fraction of parity blocks. The introduced code supports efficient updates to the file system and the sparse structure supports very large file systems.

D. Tahoe-LAFS

Tahoe-LAFS [1] is an open source distributed cloud storage system[4]. Its focus is on high availability, i.e., even if some of the cloud providers fail or are taken over by an attacker, the entire filesystem continues to function correctly, including preservation of privacy and security. The cloud provider should never have the ability to read or modify user data. Basically, their approach is to encrypt data, erasure code it and then disperse it among a set of storage providers (cf. §II-B2). Tahoe-LAFS implements access control based on cryptographic capabilities. It supports immutable files which are read only and mutable files, whereas everybody holding an appropriate cryptographic capability corresponding to a write privilege for such a file should be able to modify those files. Files are authenticated by computing Merkle-trees over the encrypted file and the file shares. In case of mutable files these roots are signed with a per-file RSA private key. Tahoe-LAFS also provide read freshness by integrating version numbers into the metadata of the shares. It requires a so-called Tahoe-LAFS gateway, which performs all of the encryption/decryption, encoding/decoding as well as integrity checking.

E. Cleversafe

In [9] the authors propose a novel information dispersal algorithm denoted as AONT-RS (see §II-B4) to be used within their distributed cloud storage system called Cleversafe. It provides both, an object- and a block-interface (to support standards like NFS, CIFS, iSCSI, etc.) for storing objects. Cleversafe requires a client, the so-called accessor, which realises all the encoding and decoding and a gateway that translates between various file protocols and iSCSI. It implements identity based access control based on access control lists (ACLs) stored along with every data slice and access control is enforced by the cloud. It also uses version numbers and transactions along with a three-phase commit protocol (parametrised via a threshold indicating how many slices have to be written to be able to conduct a commit) to provide write consistency.

F. CloudProof

The focus of the cloud storage approach in [3] is on auditability, i.e., audit the cloud that integrity, write serialisability and freshness are preserved by all operations and to detect

[4]The source code is available from http://tahoe-lafs.org/.

and proof violations to third parties. This is realised by means of signed (and chained) attestations for every operation with the cloud. It is motivated by the fact that in cloud computing there is a financial contract between clients and the cloud provider (stated as service level agreements). CloudProof requires the cloud to execute code, which can only be run in the cloud and not at some client side proxy, since it is important for auditing and has to be executed by the cloud. Indeed, their goal is to offload as much work as possible into the cloud (access control, key distribution, read, write, file creation and ensuring the security properties), but to verify the correct behaviour of the cloud. They also provide confidentiality of stored data by means of encryption as well as access control using ACLs for so-called block families (all blocks that have the same ACL belong to one block family). Furthermore, CloudProof uses broadcast encryption and key rotation [22] to reduce key management and effort necessary for re-encryption on revocation of access rights. Readers must have the read access key and writers obtain a private key of a digital signature scheme. Writes will only be accepted if the signature verification for a signature on the hash of a new block verifies. Read freshness and write-serialisability are achieved by version numbers and checking conditions for the respective attestation chains (aggregated attestations).

G. RACS

In [5] the authors propose redundant array of cloud storage (RACS), a cloud storage proxy that disperses data across multiple cloud providers by means of optimal erasure codes (cf. §II-B2). RACS only focuses on improved availability and their main motivation is to reduce the one-time cost of switching providers in exchange for additional operational overhead to prevent vendor lock-ins. Their so-called RACS proxy works with any Amazon S3 compatible client and provides adapters for other cloud storage services such as Rackspace. To avoid potential bottlenecks resulting from using a single proxy, RACS is intended to be run as a distributed system. In order to provide write consistency when clients simultaneously want to write to the same file via different proxies in this distributed setting, the single proxies coordinate their actions using one-writer, many-reader synchronisation for each stored data object.

H. NubiSave

The work reported in [10] builds upon a modular open source implementation called NubiSave[5], which provides a layered architecture (integration layer, preprocessing layer, transport layer) for a gateway to realise distributed cloud storage with a focus on easy integration of additional features, e.g., different information dispersal algorithms. The authors want to overcome a limitation in existing approaches – namely that all proposed or available distributed cloud storage systems do only implement a fixed set of algorithms, functionality and topology without configurable extensibility. A main goal of the approach in [10] is to additionally take into account arbitrary non-functional properties to support semi-automatic inclusion of new cloud storage providers into their storage pool and to provide easy to use interfaces for desktop users. They offer several methods for information dispersal, including the one presented in §II-B4.

[5]The source code is available from http://www.nubisave.org/.

I. CS2

Traditional cryptographic file systems provide confidentiality (via symmetric encryption) and integrity (via digital signatures). Although this results in strong security guarantees for the clients, it lacks functionality like efficient searching. In [8] the authors argue that file systems that do not provide search functionality are inadequate for general purpose storage systems, i.e., a file systems should be semantic. They propose CS2, a semantic cloud storage system, which integrates searchable encryption in combination with so-called search authenticators: Besides confidential search operations on encrypted data, a client should also be able to check whether the set of files returned for a search query is correct. Furthermore, their cloud storage system provides what they call global integrity, i.e., the integrity of stored data can be checked regardless of whether the entire data needs to be retrieved by a client. This is achieved via applying well known techniques from proofs of retrievability (PoRs) and provable data possession (PDP) [23] respectively. Like within IRIS, one focus is also on efficient updates of data, although this is only realised on the file level (adding and removing of entire files) without prior recovering and re-processing. But CS2 does currently not consider sharing of data between users, which is a non trivial issue in encrypted and searchable file systems.

J. Robust Data Sharing with Key-Value Stores (RDSKVS)

In [11] the authors propose an abstraction of a key-value store (KVS) as realised by many commercial cloud storage services in form of a read/write register which allows multiple clients to access data in the cloud in a multiple reader and multiple writer (MRMW) setting. They support standard operations of KVSs, i.e., store, retrieve, list and remove. Their register tolerates asynchrony, concurrency and faults among the clients as well as any minority of faulty cloud storage providers. Furthermore, in contrast to other byzantine resistant approaches, e.g., [4], [7], clients do not need to communicate. Thus, they do not need to be aware of each other and the approach is wait-free. Their approach works in the cloud of clouds setting, i.e., uses KVSs at different cloud providers, and the authors provide two different implementations: Firstly, an approach which emulates a register with regular semantics which does not need prior read operations to write data and requires to store every value in each KVS twice. Secondly, they provide an extended approach which represents an emulation of an atomic register and uses the standard approach of requiring readers to write back the value. The authors in [11] show that both of their proposed approaches have optimal space complexity, i.e., require two copies of every value per KVS in the common case. We note that

TABLE I
PROPERTIES SUPPORTED BY STATE-OF-THE-ART APPROACHES TO (DISTRIBUTED) CLOUD STORAGE SYSTEMS.

Property	DepSKY	HAIL	IRIS	Tahoe-LAFS	Cleversafe[a]	CloudProof	RACS	NubiSave	CS2	RDSKVS
Confidentiality	✓	×	×	✓	✓	✓	×	✓	✓	×
Integrity	✓	✓	✓	✓	×	✓	×	✓	✓	×
Availability	✓	✓	✓	✓	✓	×	✓	✓	×	✓
Authenticity	×	×	✓	✓	×	✓	×	×	×	×
Retrievability	×	✓	✓	×	×	×	×	×	✓	×
Freshness	✓	×	✓	✓	×	✓	×	×	×	✓
Consistency	✓	×	✓	×	✓	✓	×	×	×	✓
Access Control	×	×	×	✓	✓	✓	×	×	×	×
Auditing	×	×	✓	×	×	✓	×	×	×	×
Cloud Logic	×	✓	×	×	×	✓	×	×	✓	×
Implementation	Java	C++	C#	Python	Java	C#	Python	Python	C++	Java

[a]This is a commercial product, but its design rationale is documented in [9]. Other commercial approaches such as TrustedSafe (http://www.trustedsafe.de/) or the open source project Cumulus4j (http://cumulus4j.org) are not presented due to missing or insufficient documentation.

the purpose of the work in [11] is to design space optimal distributed cloud storage for a multi-reader and multi-writer setting without requiring communication among the clients and do not consider additional security features. Nonetheless, many of these features can be build on top of this abstract read/write register.

K. Overview of Supported Properties

Table I presents an overview of properties supported by the different approaches, which we briefly explain in the following: By *confidentiality* we mean data privacy w.r.t. (colluding) cloud providers. By *integrity* we mean measures to verify block or file integrity and do not consider using error correcting codes or erasure codes as a means to provide integrity. Although they tolerate corruptions to some degree, which then can be corrected, it is not the goal to detect whether modifications have happened. Information dispersal and consequently using the cloud of clouds paradigm is understood as a means to provide *availability*. By *authenticity* we mean additional means to check whether a file was written/modified by an authorized writer. *Retrievability* is understood in the sense of PDP [23] or PoRs [21], i.e., whether it is possible to challenge cloud providers to prove possession of a file without the necessity to be in possession of a local copy of the file for proof verification. *Freshness* and *consistency* are interesting in context of multiple-writers, i.e., whether it is possible to verify if the cloud provider(s) deliver the actual version of a file and if the approach has some means to guarantee write-consistency in the presence of multiple parallel writers to the same file respectively. The meaning of *access control* is obvious and *auditability* means whether it is possible to prove cloud violations, e.g., a dropped write operation, to a third party. *Cloud logic* means whether the cloud is required to execute code. In it is required, it is necessary to augment current cloud storage services, such as Amazon S3, by additional functionality. *Implementation* means whether a (prototypical) implementation is available and which language was used.

Note that not all properties can be treated as being independent from others. For instance, if only a single writer is considered in the design of an approach, properties like freshness and consistency are not meaningful. The same argument holds for access control, i.e., if a system is for instance designed as a backup solution to work with a single client.

IV. ACCESS PRIVACY

What has not been considered in all above mentioned approaches so far is to realise what we call *access privacy*, i.e., prevent profiling customer's usage behaviour by cloud providers. It would be desirable to achieve properties (at least to some extent) similar to those known from anonymous storage or publishing systems [24], [25]. What we mean by access privacy, in a nutshell, is that the cloud storage providers do not learn *who* accesses data or has access to data, *when* and *which* data is accessed, *what* action a client is taking or can potentially take, e.g., read, write, *how much* data is stored and from *where* a client is accessing data.

Access privacy is somewhat comparable to traffic analysis in communication networks [26]. Even if content data is encrypted, there are side channels which can leak sensitive information, e.g., identities of sender and receiver, message size, time and frequency of communication, etc. This fact led to the design of anonymous communication networks, which are an active and well established field of research. However, such side channels do also exist in storage systems. They are in part addressed in the study of censorship resistant anonymous publishing systems [24], [25]. However, such side channels are also problematic in a general setting when enterprises or private customers outsource their data into the cloud. Unfortunately, there are only few works [27]–[29] considering such side channels in cloud storage systems, typically only in part and so far not at all in the distributed cloud storage setting.

For example, cloud providers may use these side channels to learn the access history, which reveals user's habits and privileges. The fact that there exist reads to the same file from different users may indicate common interest or a collaborative relationship and the access frequency of files may also reveal individual and enterprise related interests respectively. It is known [30] that such information are confidential business information.

There are already some recent works on access privacy in the cloud setting [31]–[33] and for outsourced data in general [34]–[36]. Furthermore, recently, there is also an increased interest in making well known primitives like private information retrieval (PIR) [37] as well as oblivious RAM (ORAM) [38] practical. However, none of these issues have been considered so far in proposed architectures for distributed cloud storage. It is thus interesting for future research to investigate how current architectures work, the interplay of access privacy in distributed cloud storage as well as to study how privacy features can be efficiently integrated in current approaches.

V. Conclusion

In this paper we took a look at state of the art distributed cloud storage approaches and provide an overview of different (security) features such systems support. We have observed, that access privacy features as currently not considered in such approaches, although considered to be an important feature especially for enterprises outsourcing their data into the cloud [30]. Access privacy in a concrete (distributed) cloud storage setting is an important direction for future research.

Acknowledgments

We thank Tobias Pulls for valuable comments on a draft of this paper. This work has been supported by the Austrian Research Promotion Agency (FFG) through project ARCHISTAR, grant agreement number 832145.

References

[1] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: the least-authority filesystem," in *StorageSS*, 2008, pp. 21–26.

[2] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," in *16th ACM Conference on Computer and Communications Security, CCS '09*. ACM, 2009, pp. 187–198.

[3] R. A. Popa, J. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, "Enabling Security in Cloud Storage SLAs with CloudProof," in *USENIX Annual Technical Conference (USENIX)*, 2011.

[4] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket, "Venus: verification for untrusted cloud storage," in *CCSW*, 2010, pp. 19–30.

[5] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "Racs: a case for cloud storage diversity," in *1st ACM Symposium on Cloud Computing, SoCC 2010*, 2010, pp. 229–240.

[6] E. Stefanov, M. van Dijk, A. Oprea, and A. Juels, "Iris: A scalable cloud file system with efficient integrity checks," in *28th Annual Computer Security Applications Conference (ACSAC 2012)*. ACM, 2012.

[7] A. N. Bessani, M. P. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-of-Clouds," in *EuroSys*, 2011, pp. 31–46.

[8] S. Kamara, C. Papamanthou, and T. Roeder, "CS2: A Searchable Cryptographic Cloud Storage System," Microsoft Research, Tech. Rep. MSR-TR-2011-58, 2011.

[9] J. K. Resch and J. S. Plank, "AONT-RS: Blending Security and Performance in Dispersed Storage Systems," in *FAST*, 2011, pp. 191–202.

[10] J. Spillner, G. Bombach, S. Matthischke, J. Muller, R. Tzschichholz, and A. Schill, "Information Dispersion over Redundant Arrays of Optimal Cloud Storage for Desktop Users," in *UCC*, 2011, pp. 1–8.

[11] C. Basescu, C. Cachin, I. Eyal, R. Haas, A. Sorniotti, M. Vukolic, and I. Zachevsky, "Robust Data Sharing with Key-Value Stores," in *Intl. Conference on Dependable Systems and Networks (DSN)*. IEEE, 2012.

[12] Amazon Outage, http://www.networkworld.com/news/2012/042712-amazon-outage-258735.htmll, Nov. 20. 2012.

[13] Sidekick, http://abcnews.go.com/Business/sidekick-disaster-shows-datas-safe-cloud/story?id=8840420, Nov. 20. 2012.

[14] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage," in *CRYPTO*, 1995, pp. 339–352.

[15] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[16] P. Feldman, "A Practical Scheme for Non-interactive Verifiable Secret Sharing," in *FOCS*, 1987, pp. 427–437.

[17] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[18] H. Krawczyk, "Secret sharing made short," in *CRYPTO*, 1993, pp. 136–146.

[19] P. Rogaway and M. Bellare, "Robust computational secret sharing and a unified account of classical secret-sharing goals," in *ACM Conference on Computer and Communications Security*, 2007, pp. 172–184.

[20] R. L. Rivest, "All-or-Nothing Encryption and the Package Transform," in *Fast Software Encryption (FSE '97)*, 1997, pp. 210–218.

[21] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *ACM Conference on Computer and Communications Security*, 2007, pp. 584–597.

[22] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," in *FAST*, 2003.

[23] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *ACM Conference on Computer and Communications Security*, 2007, pp. 598–609.

[24] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Workshop on Design Issues in Anonymity and Unobservability*, 2000, pp. 67–95.

[25] M. Waldman, A. D. Rubin, and L. F. Cranor, "Publius: A robust, tamper-evident, censorship-resistant web publishing system," in *Usenix Security Symposium*, 2000.

[26] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," in *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag New York, Inc., 2001, pp. 10–29.

[27] D. Slamanig, "Efficient Schemes for Anonymous yet Authorized and Bounded Use of Cloud Resources," in *Selected Areas in Cryptography (SAC 2011)*, ser. LNCS, vol. 7118, 2011, pp. 73–91.

[28] T. Pulls, "(More) Side Channels in Cloud Storage: Linking Data to Users," in *Privacy and Identity Management for Life*, ser. AICT, vol. 375, 2012, pp. 102–115.

[29] ——, "Privacy-Friendly Cloud Storage for the Data Track: An Educational Transparency Tool," in *17th Nordic Conference on Secure IT Systems (NordSec 2012)*, ser. LNCS, 2012.

[30] Y. Chen, V. Paxson, and R. H. Katz, "What's New About Cloud Computing Security?" University of California, Berkeley, Tech. Rep. UCB/EECS-2010-5, 2010.

[31] M. Franz, P. Williams, B. Carbunar, S. Katzenbeisser, A. Peter, R. Sion, and M. Sotáková, "Oblivious Outsourced Storage with Delegation," in *Financial Cryptography*, 2011, pp. 127–140.

[32] M. Raykova, H. Zhao, and S. Bellovin, "Privacy Enhanced Access Control for Outsourced Data Sharing," in *Financial Cryptography and Data Security*, ser. LNCS. Springer, 2012.

[33] D. Slamanig, "Dynamic Accumulator based Discretionary Access Control for Outsourced Storage with Unlinkable Access," in *Financial Cryptography and Data Security*, ser. LNCS. Springer, 2012.

[34] J. Camenisch, M. Dubovitskaya, and G. Neven, "Oblivious Transfer with Access Control," in *ACM Conference on Computer and Communications Security*. ACM, 2009, pp. 131–140.

[35] J. Camenisch, M. Dubovitskaya, G. Neven, and G. M. Zaverucha, "Oblivious Transfer with Hidden Access Control Policies," in *Public Key Cryptography*, ser. LNCS, vol. 6571. Springer, 2011, pp. 192–209.

[36] S. E. Coull, M. Green, and S. Hohenberger, "Access Controls for Oblivious and Anonymous Systems," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, p. 10, 2011.

[37] F. Olumofin, "Practical Private Information Retrieval," Ph.D. dissertation, University of Waterloo, 2011.

[38] P. Williams, "Oblivious Remote Data Access Made Practical," Ph.D. dissertation, Stony Brook University, 2012.