SoK: On Cryptography for Multi-Cloud Storage

Dayane Horkos[1,2][0009−0004−9331−9359] and Ludovic Perret[2,3][0009−0005−7453−8705]

[1] Astran, Paris, France, dayane.horkos@astran.ai
[2] EPITA, EPITA Research Lab (LRE), Le Kremlin-Bicêtre, France, ludovic.perret@epita.fr
[3] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Abstract. This paper is a Systematization of Knowledge (SoK) on cryptography applied in Multi-Cloud Storage (MCS) schemes. Such techniques distribute and fragment data among multiple cloud providers to strengthen confidentiality, integrity, and availability compared to single-cloud deployments. Over the past decade, many cryptographic mechanisms have been proposed to secure outsourced data. However, the lack of unified framework has led to fragmented terminology, inconsistent trust assumptions, and unclear comparisons between existing protocols. In this SoK, we analyze and classify existing MCS approaches along two complementary dimensions: (*i*) their architectural trust models, and (*ii*) the cryptographic primitives they employ to achieve confidentiality, integrity, availability, access control, and auditability. This taxonomy allows us to highlight recurring design patterns, evaluate trade-offs between security and efficiency, and identify redundancies and research gaps. By unifying terminology and mapping the landscape of cryptographic techniques in MCS, this SoK provides both researchers and practitioners a structured foundation to guide future designs and deployments.

Keywords: SoK · Multi-Cloud Storage · Cryptography · Cryptographic Protocols.

1 Introduction

With the growing demand for data storage, cloud solutions have become the default choice for individuals and enterprises alike, thanks to their flexibility, scalability, and low cost. However, relying on a single cloud provider raises well-known concerns about data confidentiality, integrity, and availability [93,29]. For example, outages in major providers such as AWS, Microsoft Azure, or Alibaba Cloud [48], and other real world incidents reported in [105], have repeatedly shown that entrusting data to a single entity creates a critical single point of failure and risks of vendor lock-in. As emphasized in [4], clouds should not be treated as secure vaults for cryptographic secrets, since this corresponds to sharing ownership of the underlying assets with the provider.

These risks have motivated a shift towards multi-cloud (also called cloud-of-clouds or intercloud [107]), where services are distributed across multiple independent providers. From a research perspective, the concept of utilizing multiple cloud environments has evolved significantly over the past decade. Early visions framed a "cloud of clouds" as a loosely interconnected ecosystem inspired by the Internet; this concept evolved into federated cloud models [27] and intercloud frameworks with standardized protocols and fault-tolerant coordination [109,118]. In practice, systems such as RACS [3] pioneered distributing data across multiple cloud vendors to improve availability, resilience, and reduce vendor lock-in. This paradigm has moved from research to practice, with platforms like Google Anthos, Astran, Vawlt, and MultCloud enabling fragmentation, dispersion, and orchestration across heterogeneous infrastructures.

Today, Multi-Cloud Storage (MCS), broadly refers to splitting and allocating partial data across several providers to balance security, availability, cost, and performance. For instance, Yang et al. [118] model share allocation among providers as a multi-objective optimization problem that trades off trust-based security, access time, and cost. Most secure MCS schemes achieve this by fragmenting, dispersing, and cryptographically protecting data shares prior to storage. Indeed, the only robust way to prevent unauthorized access or misuse is to make such actions cryptographically impossible without possession of specific secrets, and to withhold those secrets from untrusted parties.

Despite rapid deployment of MCS, e.g. [90,17,75,117,108,28,2,66,46,116,97,19,68], the academic literature remains fragmented. As Alqahtani et al. [7] note, there are still no standard definitions or classifications that capture the features of existing MCS models. This lack of consensus motivates the need for a systematization of knowledge (SoK) that connects architectural trust models with the cryptographic foundations that underpin security guarantees.

1.1 Related Works

Several surveys and reviews have examined cloud and multi-cloud security. Below, we position our SoK within this landscape.

Single-cloud security surveys. Papers such as [47], which reviews threats and cryptographic countermeasures for cloud storage (e.g. encryption, hashing), or [51], which analyzes client-side cryptographic defenses (e.g. encryption, PDP, POR, access control), remain insufficient because they address only single-provider architectures which risk vendor lock-in.

One primitive–focused review. Since the core challenge is securely splitting data among multiple parties, secret sharing has been extensively studied as a basis for confidentiality, availability, and integrity in cloud storage [35]. Likewise, [11] surveys variants of Shamir's Secret Sharing and their applications to privacy and secure computation. In addition, [41] explores All-Or-Nothing Transforms (AONTs) for distributed storage, and [14] examines erasure coding for fault tolerance. While valuable, such works focus on isolated primitives, whereas MCS schemes integrate and combine multiple cryptographic mechanisms by finding a trade-off between security and efficiency.

Multi-cloud architectures and surveys. The security of MCS has garnered significant attention in the academic community. For example, AlZain et al. [10] provide an early survey that motivates the transition to multi-cloud to mitigate insider threats, outages, and vendor lock-in, and reviews initial MCS systems such as RACS, HAIL, DepSky, and ICStore. Other conceptual works, e.g., Kathuria et al. [57] and Tupe [107], further motivate multi-cloud through replication and integrity verification. Bohli et al. [22] offer one of the first structured taxonomies of multi-cloud architectures, distinguishing replication of applications, partitioning of tiers, partitioning of logic, and partitioning of data, and analyzing trade-offs among confidentiality, integrity, availability, and compliance. Other overviews, such as [44,49], focus primarily on system-level aspects like resource provisioning, elasticity, and performance. These works remain architectural and do not systematize the underlying cryptographic building blocks for storage.

Cryptography in multi-clouds. Closer to our focus, Naik and Kannan [53] reviewed cryptographic techniques in distributed and multi-cloud settings. They proposed the Multi-Cloud Security Technique (MCST), but their evaluation was limited to performance comparisons. The protocol SeDaSC [105] discusses attribute-based encryption and proxy re-encryption for multi-cloud data sharing, but does not extend to the wide range of primitives used in storage protocols. Beyond surveys, Kapusta's PhD thesis [54] explores efficient keyless fragmentation algorithms for secure storage, highlighting performance-oriented dispersal strategies, and Ukwandu's thesis, RESCUE [106], is limited to data sharing methods in its design methodology and does not discuss architectural constructions neither the presence of a proxy. In addition, Shor et al. [98] conduct a comparative performance study of encryption and secret-sharing–based confidentiality mechanisms in multi-cloud environments, analyzing schemes such as AONT-RS, and secure RAID techniques. Their evaluation remains performance-oriented rather than taxonomic. These contributions offer valuable insights, but they do not provide a unified systematization of MCS protocols across trust models and security goals, which remains the focus of our SoK.

Limitations of prior surveys. In summary, while the security of multi-cloud storage has garnered increasing attention, existing surveys either *(i)* focus on single-cloud threats, *(ii)* restrict themselves to a single primitive, *(iii)* analyze architectures without systematizing the cryptographic techniques underpinning MCS protocols, or *(iv)* discuss cryptography in multi-cloud without giving a unified systematization. Table 1 compares this SoK with the previous works. We include the most representative papers from the state of the art; additional references can be found in the relevant sections.

Table 1. Positioning of related work.

References	Type	Cloud computing	Multi-cloud architecture	Cryptography	Main contribution
[47], [51]	Survey	●	○	◍	Architecture of a cryptographic storage service
[11], [35]	Survey	◍	◍	●	Secret sharing and its variants
[41]	Thesis	◍	◍	●	AONTs and their applications for secure distributed storage
[14]	Survey	◍	◍	●	Erasure coding in MCS
[10]	Survey	●	◍	◍	Early survey of cloud storage schemes and limitations
[107], [57]	Survey	●	●	○	Early motivation for multi-cloud approach
[22]	Survey	●	●	◍	Architectural taxonomy of multi cloud storage patterns
[44], [49]	Survey	●	●	○	System-level aspects of multi cloud schemes
[53]	Survey	●	●	◍	System architecture over multi clouds with some focus on cryptography
[105]	Survey	●	◍	●	Data sharing with Attribute based encryption and Proxy reencryption
[54]	Thesis	◍	◍	●	Keyless fragmentation algorithms for secure distributed storage
[106]	Thesis	◍	●	◍	Data sharing methods in MCS architecture
[98]	Evaluation	●	◍	●	Performance-oriented comparison of MCS
Our work	SoK	●	●	●	Taxonomy on distributed storage by trust model and cryptographic primitives

Legend: ● substantial coverage ◍ partial coverage ○ not addressed

1.2 Our Contributions

In this SoK, we systematize the literature on Multi-Cloud Storage (MCS) along two complementary dimensions to provide a structured analysis. First, at the architectural level, we classify schemes according to their trust model, distinguishing between systems relying on a trusted proxy (or no proxy), and systems employing an untrusted proxy, and in each case, we further differentiate between keyless and key-leveraging protocols. Second, at the cryptographic level, we analyze the techniques used to achieve the core security objectives of confidentiality, availability, and integrity, while

also covering additional properties such as access control and retrievability. This two-dimensional taxonomy enables us to: *(i)* highlight recurring design patterns, *(ii)* evaluate trade-offs between security, efficiency, and deployability, and *(iii)* identify trends, redundancies, and open problems in the current landscape.

To the best of our knowledge, this is the first SoK that offers a complete picture of the cryptographic concepts already deployed — or with strong potential to be deployed — in MCS. We do not claim to exhaust all cryptographic primitives and protocols explored in this context; rather, we focus on those we consider most relevant and beneficial to the scope of this paper.
Beyond surveying existing works, our goal is twofold: *(i)* to provide a unified terminology and framework for reasoning about MCS security, and *(ii)* to guide practitioners and system designers in selecting appropriate cryptographic mechanisms and architectures according to their deployment requirements and security needs.

In summary, our work provides a comprehensive SoK that not only surveys existing MCS schemes but also bridges their underlying cryptographic foundations. By connecting architectural trust models with secure primitives, analyzing how different techniques combine to achieve confidentiality, availability, and integrity, and incorporating state-of-the-art advances, we deliver a resource that informs future research and aids practical deployment of secure multi-cloud storage.

1.3 Organization of the Paper

The remainder of this paper is structured as follows. Section 2 presents a systematization from the MCS perspective, introducing our system model, design goals, trust-based taxonomy, and key management models, as well as the security objectives typically required in these settings. Section 3 surveys the cryptographic primitives underlying MCS schemes, organized according to the security goals they support. Section 4 systemizes existing multi-cloud storage protocols, classifying them according to the assumptions defined earlier, and analyzing their trade-offs, recurring patterns, and open challenges. Finally, Section 5 summarizes the key findings and insights of this study.

2 System Model and Design Goals

2.1 Multi-Cloud Storage

A Multi-Cloud Storage (MCS) scheme involves three entities interacting as shown in Figure 1.

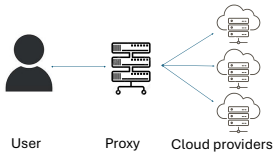


Fig. 1. Multi-Cloud Storage components.

1. *User*. An honest entity that wishes to store data securely on multiple clouds. This entity represents a logical role that may encompass data owners who upload data and authorized users who may access or retrieve stored data according to access control policies.
2. *Proxy (optional)*. An optional intermediary responsible for managing cloud infrastructures, including selecting cloud providers, handling the Disperse operation, and coordinating the distribution of data across the storage network.
3. *Cloud Service Providers (CSPs)*. n independent cloud providers who Receive the corresponding shares from the user or proxy and store them.

There are two phases: *upload*, where the user stores data $\mathcal{D}$ via the proxy (if present), and *download*, where the user retrieves shares from any subset of at least k CSPs and reconstructs $\mathcal{D}$.

2.2 MCS Assumptions

Trust Assumptions. Based on our research, we group MCS protocols into two main trust models that form the backbone of our taxonomy.

- *Trusted processor (user or trusted proxy)*. All cryptographic operations (which aim to protect confidentiality, availability, integrity, etc) are performed by a fully trusted entity. This entity may be the user himself (no proxy) or an external trusted proxy. It has access to plaintext data, and is trusted to behave honestly.
- *Untrusted proxy*. A middle proxy delegated the performance of heavy data processing and distribution, but is *not* trusted with plaintext or secret keys. It must be constrained by cryptographic mechanisms to prevent data exposure or tampering.

Key Management Assumptions. In addition to the trust placed in the proxy, MCS protocols differ in how they manage cryptographic secrets. We therefore distinguish two complementary key management models that cut across both trust settings.

- *Key-leveraging protocols*. These schemes assume that the trusted processor retains secret material — such as encryption keys, long-term hashes, or identity private keys.
- *Keyless protocols*. Also referred to as forgetful user models [60], where no long-term secrets are stored between upload and download by the trusted entity that handles the Process operation. During retrieval, the user may rely on transient or ephemeral secrets (e.g., short-lived tokens) to decrypt or reconstruct data.

These models determine where cryptographic responsibilities lie, and fundamentally shape how confidentiality, integrity, and availability are enforced.

2.3 Abstract MCS Interface

Based on these assumptions, we generalize a Multi-Cloud Storage (MCS) scheme with the following design goal: to modularize data preparation, distribution, and reconstruction, allowing flexibility in how roles are assigned to entities. As by the name, the system uses a single type of service provided by the CSPs: storage. Taking into account the previous literature, we define MCS as follows.

Definition 1. *A Multi-Cloud Storage (*MCS*) scheme is a tuple of probabilistic polynomial-time (ppt) algorithms,* $\Pi_{\mathsf{MCS}} = (\mathsf{Param}, \mathsf{Process}, \mathsf{Disperse}, \mathsf{Receive}, \mathsf{Retrieve}, \mathsf{Merge}, \mathsf{Rebuild})$*, that operates on data* $\mathcal{D}$*, where each function operates over system parameters and adapts based on the trust relationship between the user, the proxy, and the CSPs.*

Global Parameters

- $\mathcal{D} \in \{0,1\}^*$*: the original data to be stored.*
- $Param \rightarrow (\lambda, n, k)$*:* λ *is the security parameter and* $(k, n) \in \mathbb{N}^2$ *are the threshold parameters, where any* k *of* n *total fragments suffice for recovery. The threshold may range from minimal* $(k < n)$ *to full threshold* $(k = n)$ *depending on the desired confidentiality-availability trade-off.*

Upload Phase

- $\mathsf{Process}(\mathcal{D}, Param) \rightarrow (\mathcal{D}', \{s_i\}_{i=1}^n)$*: Transforms the input data* $\mathcal{D} \in \{0,1\}^*$ *into a processed form* $\mathcal{D}'$ *and auxiliary elements* $\{s_i\}$*. The* s_i *may include encrypted shares, MACs, signatures, or metadata depending on the scheme.*
- $\mathsf{Disperse}(\mathcal{D}', \{s_i\}_{i=1}^n, Param) \rightarrow \{c_i\}_{i=1}^n$*: Takes the processed data and outputs storage-specific payloads* c_i *destined for the cloud providers. Encapsulates redundancy and recoverability through secret sharing, erasure coding, or encryption.*
- $\mathsf{Receive}(c_i, aux_i) \rightarrow store_i$*: Defines how each CSP* i *handles the incoming data fragment* c_i *along with optional auxiliary metadata.*

Download Phase

- $\mathsf{Retrieve}(i) \rightarrow (c_i, aux_i)$*: Cloud provider* i *returns the stored fragment* c_i *along with auxiliary information* aux_i *(e.g., integrity metadata, encryption context, or audit proof).*
- $\mathsf{Merge}(\{c_i\}_{i\in I}, Param) \rightarrow (\mathcal{D}', \{s_i\}_{i\in I})$*: Takes any subset* $I \subseteq \{1, ..., n\}$ *with* $|I| \geq k$ *of returned fragments and reconstructs the intermediate processed data* $\mathcal{D}'$ *and associated elements* $\{s_i\}$*.*
- $\mathsf{Rebuild}(\mathcal{D}', \{s_i\}) \rightarrow \mathcal{D} \cup \perp$*: Recovers the original data* $\mathcal{D} \in \{0,1\}^*$ *from the merged view, or outputs* $\perp$ *in case of failure (e.g., due to corruption or insufficient fragments).*

2.4 Security Goals

Depending on the application and the protocol, the following security properties may be required.

- *Confidentiality.* No information about $\mathcal{D}$ is leaked to unauthorized sets of users.
- *Integrity.* Any modification of stored fragments is detectable during recovery.
- *Availability.* Data remains recoverable as long as at least k valid fragments remain accessible.
- *Accountability.* Faulty or malicious behavior can be traced and attributed.
- *Access Control.* Only authorized entities can get access to the data.
- *Retrievability / Auditability.* The system can prove that data is retrievable and has not been lost.

These security properties can vary depending on the tradeoffs that should be made. It is important to take into consideration both the computational security and the space efficiency parameters when implementing these solutions.
Among other properties, we can find Fork Consistency (users detect if a malicious CSP presents divergent views of the data [64,73]) and Authenticity, which allows users to verify the origin of the recovered data.

In multi-cloud settings, security guarantees are typically required to hold even when a subset of up to f cloud providers behaves arbitrarily or maliciously (i.e., under Byzantine faults). Rather than constituting a separate security goal, Byzantine Fault Tolerance $\mathsf{BFT}(f)$ characterizes the

adversarial conditions under which confidentiality, integrity, availability, and accountability must be preserved, assuming a total of n providers and appropriate resilience bounds (e.g., $n \geq 3f + 1$).

Furthermore, the paper [24] introduces formal security notions for cloud storage which we do not discuss here. Other common threats in MCS environments include: data exposure or unauthorized access, data loss or manipulation, service level agreement (SLA) violations, malicious insiders at cloud providers, insecure APIs, shared-technology vulnerabilities in multi-tenant infrastructure, and weak registration processes enabling attacker-controlled accounts. We assume data is transmitted securely using encrypted channels such as TLS to prevent interception.

3 Cryptographic Primitives used in MCS Schemes

In this section, we discuss the cryptographic primitives used in MCS schemes and we classify them based on their main security goals.

3.1 Confidentiality Primitives

We define confidentiality primitives as the building blocks that provide secrecy in an MCS scheme. The primary goal of any MCS scheme is to protect user data against untrusted proxies and cloud providers. This requires decoupling confidentiality from storage and placement decisions, ensuring that data secrecy holds regardless of how and where the data is stored. Confidentiality primitives are usually used in the Process phase of the upload (Definition 1).

Encryption. The most common method for ensuring confidentiality in MCS is encryption. The canonical choices for securing large volumes of data are Secret-Key Encryption (SKE) and Public-Key Encryption (PKE) schemes. Typical SKE algorithms include AES used with different modes of operation. For instance, CBC provides confidentiality only (and requires a separate mechanism for integrity), whereas AEADs modes such as EAX (or GCM) provide both confidentiality and integrity/authenticity, making them well-suited for modern cloud storage systems.
Earlier works also employed legacy ciphers such as 3DES; however, such constructions are no longer considered suitable for contemporary deployments due to efficiency and security limitations.

In the literature, the central challenge for SKE-based systems has consistently been the secure management and protection of the Data Encryption Key (DEK). In hybrid designs, PKE is often employed to protect these DEKs—typically through a Key Encapsulation Mechanism (KEM) that encapsulates a symmetric key, combined with a Data Encapsulation Mechanism (DEM) that encrypts the data itself [99].

Beyond conventional SKE–PKE hybrids, more advanced cryptographic approaches have been explored to enhance confidentiality and flexibility, including proxy re-encryption and homomorphic encryption.

- *Proxy Re-Encryption (*PRE*).* This concept is very appealing for MCS [21] because it enables a data owner to delegate the decryption of the encrypted data to the authorized recipients by passing through a proxy. A semi-trusted proxy transforms a ciphertext for key A into one for key B without learning the plaintext. Certificate-based proxy re-encryption [71] is a new cryptographic primitive to effectively support the data confidentiality in public cloud storage, which enjoys the advantages of certificate-based encryption while providing the functionalities of proxy re-encryption.

– *Homomorphic Encryption (*HE*).* Another interesting concept in MCS [43] because it enables computations on encrypted data across different providers without revealing the underlying plaintext, thus preserving confidentiality even during processing. For example, [60] used HE to enable secure delegation of share generation and distribution to an untrusted proxy, eliminating the need for user-side keys while keeping data confidential across multiple clouds.

All-Or-Nothing Transform. Introduced by Rivest in 1997 [92], an All-Or-Nothing Transform (AONT) is a keyless transformation that ensures no partial information about the input is accessible unless the entire output is available. Originally designed to increase resistance to brute-force attacks on symmetric encryption, AONTs have since been used as lightweight confidentiality layers in storage systems. A useful generalization of an AONT was introduced by [30], who suggested considering an AONT which has a two-part output: a public output that can be revealed to the adversary (similar to an encrypted data), and a secret output that has the exposure-resilience property and needs to be protected (similar to a DEK).

Definition 2. *An* AONT *is a randomized transformation* $f : \{0,1\}^* \to \{0,1\}^*$ *such that:*

– *For* $x_i \in \{0,1\}^\lambda, i = 0,\ldots,n$*, given the entire output* $f(x_0\|\cdots\|x_n) = d_0\|\cdots\|d_\ell$*,* $d_j \in \{0,1\}^\lambda, j = 0,\ldots,\ell$*, it is easy to recover the input* $\mathcal{D} = x_0\|\cdots\|x_n$*.*
– *Given all but one block of the output (the secret output), i.e., any* $\{d_j\} \in \{0,1\}^\lambda, j = 0,\ldots,\ell$ *missing, it is computationally hard to learn anything about any* $x_i \in \{0,1\}^\lambda, i = 0,\ldots,n$*.*

Among AONTs presented in the literature, we discuss those used to provide confidentiality in MCS.

1. AONTRP. Resch and Plank [90] introduced an AONT variant (we refer to it as AONTRP), which hides the data by encrypting it under a random key and binding the key to the ciphertext via a hash. Multiple variations of this AONT have been proposed in the literature. The secure version SAONT [120] adds stronger tags for short-plaintext security, while the convergent instantiation CAONT [65] adapts the transform for deduplication in MCS.
2. OAEP. Optimal Asymmetric Encryption Padding [16], widely used in RSA encryption, can also be viewed as an AONT. Its randomized encoding ensures that partial exposure yields no information. Boyko shows that OAEP satisfies strong AONT security notions (formalized via polynomial indistinguishability) in the random oracle model [25]. A convergent variant of OAEP has been adopted in MCS schemes such as CDStore [66].

Karame et al. [56] introduced bastion AONTs, as an extension of Stinson's information-theoretically secure AONTs [101]. This construction is fast and efficient but not keyless, it was designed to protect distributed encrypted data when the attacker has access to the encryption key. Other more general AONT constructions were presented by [41] in the context of distributed storage, with variations depending on the assumptions made about the input and output blocks.
From a practical perspective, AONTs occupy an intermediate position between encryption and secret sharing. Unlike encryption, AONTs do not rely on a long-term secret key and therefore avoid key-management overhead; however, their confidentiality guarantees rely on the assumption that at least one output block remains unavailable to the adversary. Unlike secret sharing, AONTs incur minimal storage overhead and are computationally lightweight, but they do not tolerate the permanent loss of output blocks. As a result, AONTs are often combined with erasure coding to balance confidentiality, availability, and efficiency.

3.2 Confidentiality and Access Control Primitives

Some primitives, which are also usually used in the Process phase, simultaneously enforce confidentiality and fine-grained access control; we discuss them separately here.

1. *Attribute-Based Encryption (*ABE*).* It is a cryptographic primitive that enables fine-grained access control over encrypted data. Decryption succeeds only if the attributes associated with a user's secret key satisfy the policy embedded in the system.
   In CP-ABE (Ciphertext-Policy ABE), the user defines an attribute-based access policy which is embedded into the ciphertext. In MCS, CP-ABE eases data sharing among cloud providers, as a user can specify whom they share the file with directly within the file using the access policy [121,42].

   In contrast, KP-ABE (Key-Policy ABE) reverses this mapping by embedding access policies into users' keys rather than into the ciphertext [77].

   MA-ABE (Multi-Authority ABE), removes the single point of trust by distributing attribute issuance across multiple authorities. This is especially important in multi-organizational scenarios, where no single entity should hold complete control over access management [77].
2. *Identity-Based Encryption (*IBE*).* It is a public-key encryption scheme in which public keys are arbitrary strings, such as email addresses, user roles, or time-bound tags. A Private Key Generator (PKG) issues secret keys corresponding to these identities. In MCS, IBE can be used to encrypt sensitive components directly to a user's identity, simplifying key management. However, classical single-authority IBE schemes inherently exhibit a key escrow property, as the PKG can derive all users' private keys, which is undesirable in untrusted cloud settings. Variants based on distributed PKGs have been proposed to mitigate this limitation.
   An extension of IBE is IBBE (Identity-Based Broadcast Encryption), where a symmetric secret is encrypted once to a set of identities (a group of authorized users), and each participant can independently decrypt using their identity-derived key [100].
3. *Multi-Key Encryption (*MKE*).* It allows a ciphertext encrypted under multiple public keys to be jointly decrypted only if all corresponding secret keys participate, thereby requiring collaborative cooperation for successful decryption [60].

The importance of using both ABE and PRE in MCS was discussed in detail in [105].

3.3 Confidentiality and Availability Primitives

This part focuses on cryptographic primitives that simultaneously ensure confidentiality and availability, typically through distributed trust. While such primitives offer strong guarantees, they are generally too expensive (computationally or storage) to be applied to entire datasets. Instead, they are most often used to protect small but critical components (and thus used in the Process phase) [98,8,70], such as DEKs or secret AONT outputs, which must remain both secret and recoverable even in the face of partial compromise or cloud outages [81]. When secure local storage is not feasible, distributing these secrets across multiple clouds using threshold mechanisms becomes a robust alternative.

Threshold Secret Sharing. A (k, n)-threshold secret sharing scheme distributes a secret into n shares, such that any k suffice to reconstruct the secret (*recoverability*), while any subset of fewer than k reveals negligible or no information (*privacy*).

Secret sharing can be broadly classified into perfect secret sharing (PSS), which offers information-theoretic privacy—unauthorized subsets learn no information about the secret—and computational secret sharing (CSS), which ensures privacy under standard cryptographic assumptions, allowing only negligible leakage.

In practice, most PSS-based MCS systems rely on a small set of well-established constructions. Shamir's secret sharing scheme (SSS) [95], based on polynomial interpolation, is by far the most widely used and commonly employed in practice (e.g., SSMS [59]) to protect DEKs. An alternative family of PSS schemes is based on the Chinese Remainder Theorem (CRT), such as the Asmuth–Bloom and Mignotte schemes [87], which offer different algebraic instantiations with comparable security guarantees. Blakley's scheme [20], which relies on hyperplane geometry, is less efficient and rarely used in practice.

Beyond these canonical constructions, a number of variants have been proposed to address specific optimizations or architectural concerns. Examples include PRESS [78], a recursive variant of Shamir's scheme that reduces storage overhead, and LASS (Latin-Square-Autotopism Secret Sharing) [117], which derives deterministically constructed keys to improve key management.

Secret sharing schemes offer strong security guarantees, but they incur high storage and computational costs, since each share is typically the size of the original secret. Consequently, they are applied in MCS primarily to protect small but critical secrets, while broader data availability is usually ensured via more efficient primitives such as information dispersal algorithms (discussed in the next section).

Secret sharing schemes can be generalized by Ramp Schemes [20] by introducing a tunable privacy threshold. A (ℓ, k, n) ramp scheme guarantees that: (1) the secret can be reconstructed from any k or more shares, and (2) any set of ℓ or fewer shares reveals no information about the secret. For sets of shares between ℓ and k, partial information leakage may occur, making ramp schemes suitable for applications where strict secrecy can be relaxed in favor of reduced share sizes or improved performance. These schemes interpolate between the two extremes:

- When $\ell = k$, we recover a standard perfect threshold secret sharing scheme.
- When $\ell = 0$, the scheme behaves like an information dispersal algorithm (IDA) with no secrecy guarantees.

3.4 Availability Primitives

Ensuring availability in MCS typically requires introducing redundancy, so that data remains accessible even in the presence of outages or failures. This is distinct from threshold secret sharing, which also guarantees availability but does so through distributed trust rather than redundancy. Redundancy can be achieved in different ways, the most common being replication, parity, and erasure codes. A comparison of the main cryptographic primitives used to guarantee availability in MCS (PSS and IDA) is provided in Table 2.

Replication, Parity and RAID. Replication and parity-based schemes are fundamental techniques to ensure availability in storage systems. Replication duplicates data across multiple nodes or cloud providers, allowing retrieval as long as one copy remains accessible; it provides fast recovery and minimal decoding overhead but incurs significant storage costs. Parity-based methods, such as those used in RAID systems, achieve fault tolerance more efficiently by storing parity blocks—often computed as XOR combinations of data blocks—to enable error detection without full duplication.

Originally designed for local disks, these mechanisms inspired early multi-cloud storage schemes, where parity offered lightweight redundancy with lower overhead than full replication.
More advanced approaches generalize these ideas by adding error correction into *information dispersal algorithms* (IDAs), which we discuss in the following section. Weatherspoon and Kubiatowicz [111] showed that erasure-coded storage achieves far greater durability than replication with roughly an order of magnitude less storage and bandwidth, motivating the adoption of coding-based redundancy in modern multi-cloud systems.

Information Dispersal Algorithms. To achieve stronger trade-offs between reliability and efficiency, Information Dispersal Algorithms (IDAs) were introduced, underpinned by erasure coding [84]. IDAs split data into n fragments such that any k suffice for reconstruction, offering configurable resilience with much lower storage overhead than replication. In MCS, they are particularly suited for distributing large datasets where confidentiality is ensured by other means, since IDAs by themselves do not provide secrecy as discussed in the previous section.
The first IDA was proposed by Rabin [88], and many variants have since been developed. Key families include:

- *Reed–Solomon (RS) Codes [90].* Classical Maximum Distance Separable (MDS) codes, guaranteeing recovery from any k fragments and achieving optimal storage efficiency for a given level of fault tolerance. Systematic RS codes are widely deployed; while partial collections of fragments may reveal linear information about the data (often described as a linear ramp leakage profile [36]).
- *Cauchy-Reed–Solomon (CRS) Codes [61].* An optimized RS variant replacing expensive finite-field multiplications with XOR operations, practical for storage systems.
- *Luby Transform (LT) and Raptor Codes [31,15].* Beyond classical MDS codes, near-optimal erasure codes such as Tornado or Raptor codes reduce encoding cost at the expense of needing slightly more than k (i.e., $(1+\epsilon)k$) fragments for decoding. Rateless codes (LT, Raptor) are particularly suited for streaming or unpredictable environments, though with more complex decoding. They trade MDS guarantees for flexibility and low encoding cost, making them attractive in cooperative or large-scale MCS settings.

Beyond these, High-Speed Resilient Coding (HSRC) [68] and Privacy-Protecting Codes (PPC) [97] target fast encoding and repair but offer weak confidentiality; Zigzag (NZZD) codes [115] reduce decoding/repair complexity; XOR-based designs (e.g., WEAVER [1]) improve repair efficiency with reduced storage efficiency. In addition, LWE-based proposals [119] explore post-quantum dispersal using lattice-based constructions. Unlike classical IDAs, these schemes integrate confidentiality directly into the dispersal process by relying on the hardness of the Learning With Errors problem, while simultaneously providing availability through coding-based redundancy. We discuss them here since their primary mechanism is data dispersal for fault tolerance.
To curb repair bandwidth, regenerating codes [40] use network coding to minimize transfer during node repair — NCCloud [46] is a practical example. A recent work optimizes code parameters and placement across providers [45].

Another coding approach employed in multi-cloud storage is based on Redundant Residue Number Systems (RRNS). It splits data into residues with respect to multiple pairwise coprime moduli and introduces redundancy by adding extra moduli. This enables both error detection and correction in case some residues are corrupted or lost, thereby strengthening availability and even integrity.

In MCS settings, RRNS has been combined with modular secret sharing [104] and hybrid storage systems [33] to provide resilience against malicious providers while maintaining efficient reconstruction.

Finally, coding techniques are often combined with other mechanisms — such as auditing, redundancy monitoring, and intruder detection — to provide a complete availability layer.

Table 2. Comparison between PSS and IDA.

Property	PSS	IDA
Trait	Information-theoretic confidentiality	Space efficiency
Purpose	Distribute shares of a secret	Disperse data blocks
Main methods	Polynomial interpolation, CRT, etc.	Erasure coding (e.g. RS codes)
Share size	$\lvert\mathcal{D}_i\rvert = \lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}_i\rvert = \lvert\mathcal{D}\rvert/k$
Representative schemes	Shamir's (SSS) [95]	Rabin's IDA [88]

3.5 Integrity Primitives

To ensure that outsourced data has not been altered or tampered with, MCS schemes integrate cryptographic mechanisms for integrity verification. These primitives provide assurance that the data retrieved is exactly what was originally stored. The most common approaches include Message Authentication Codes (MAC) used in symmetric-key settings, and Digital Signature Algorithms (DSA), such as EdDSA, used in public-key settings.

Beyond these standard primitives, several lightweight techniques can also strengthen integrity guarantees. Examples include embedding canaries (known check values) inside the data [90], storing and verifying cryptographic hashes of data blocks or slices [66], and re-checking consistency by attempting reconstruction with $k+1$ shares instead of exactly k in threshold-based schemes [97]. Such approaches provide simple yet effective safeguards that complement stronger cryptographic mechanisms.

3.6 Integrity and Auditability Primitives

Integrity and auditability primitives enable the client to detect any data corruption or tampering by cloud providers and to verify that the stored data is still retrievable without downloading it in full. These primitives typically require the client to embed additional verification metadata into the data before outsourcing it, and then periodically challenge the servers to produce cryptographic proofs of correct storage. The three main classes are:

- *Proofs of Data Possession (*PDP*) [23].* Allow a client to verify that a server still possesses the original data blocks without retrieving them, by checking the correctness of responses to random spot-check challenges.
- *Proofs of Retrievability (*POR*) [23].* Extend PDP by ensuring not only that data blocks are present, but also that the entire file can be reconstructed with high probability from the challenged blocks.
- *Blockchain-based Protocols.* Recent approaches leverage smart contracts and distributed verifiers to remove reliance on a single third-party auditor (TPA). For example, Witanto et al. [113] use ZSS signatures and bilinear pairings for block-level proofs, with verification tasks distributed across multiple auditors coordinated through blockchain. This ensures transparency, batch verification, and stronger auditability guarantees in multi-cloud settings.

Such techniques are widely used in MCS schemes to provide strong integrity and retrievability guarantees without compromising efficiency.

Beyond the basic PDP and POR classes, several refinements have been proposed for multi-cloud settings. For example, Blockchain-Assisted Multi-Copy PDP [76] and Identity-Based Multi-Copy PDP [63] extend PDP to scenarios where multiple replicas of data are distributed across clouds.
In addition, several dedicated public auditing protocols enable integrity verification by external entities. A common setting introduces a trusted TPA that performs verification on behalf of the client. For instance, Wang et al. [96] propose a dynamic auditing scheme based on BLS homomorphic authenticators, allowing the TPA to verify data integrity without accessing its content. While such protocols provide strong auditability guarantees and relieve clients from performing integrity checks, they rely on pairing-based cryptography, which incurs non-trivial computational costs and is not considered post-quantum secure.

3.7 Integrity, Retrievability, and Accountability Primitives

Beyond ensuring that outsourced data remains intact and recoverable, different cryptographic primitives extend it to accountability by enabling attribution of misbehavior. While PDP and POR detect data loss or tampering, they do not by themselves provide binding evidence of which party misbehaved. Accountability requires primitives that allow such attribution:

- *Commitment Schemes.* Let a storage provider commit to specific data or metadata in a binding and verifiable way, ensuring any deviation can be cryptographically proven.
- *Blockchain Audit Logs.* As in [117], recording audit results on-chain enables transparent accountability, since misbehavior can be publicly verified and traced back to the responsible provider (Nap Checks).

3.8 Byzantine Fault Tolerance (BFT)

While redundancy, coding, and cryptographic proofs ensure data availability and integrity, they typically assume that most providers behave honestly. To resist fail-stop events or even active attacks without service interruption or loss of data integrity, secret sharing and other primitives should be combined with Byzantine agreement protocols [109,38]. Consensus protocols such as PBFT [32] or other quorum-based systems [72] guarantee consistency and liveness as long as a threshold of providers are honest, while SUNDR [73], introduced fork consistency for detecting equivocation in untrusted storage. While not a cryptographic primitive per se, BFT complements MCS security goals by ensuring consistency and availability in adversarial settings, often relying on digital signatures or threshold signatures to authenticate protocol messages.

4 MCS Protocols

In this part, we survey the state of the art across the main categories of trusted and untrusted proxy models, emphasizing how different works combine cryptographic primitives to achieve their respective security goals under their security model's trust assumptions.

The first subsection assumes the presence of a trusted processor, either the user himself or a trusted proxy, which represents the most extensively studied setting in the literature. We divide it into

two parts: keyless and key-leveraging protocols. Within the keyless category, we further distinguish three categories: keyless by construction, keyless by separation, and keyless by masking. We then discuss key-leveraging protocols, which rely either on long-term secret keys, or on identity keys.

The second subsection explores systems involving an untrusted proxy, a model that remains less explored yet particularly relevant for practical MCS settings. We also divide this part into key-leveraging and keyless protocols and discuss the ambiguity often found in their underlying trust and security assumptions, proposing directions for extending them toward more realistic untrusted-processor settings. We summarize the advantages and limitations of each family.

As our SoK reveals, the majority of works assume a trusted processor; only a handful explore untrusted proxy models. Table 3 presents a comparative overview of representative protocols, where we specify for each scheme whether a proxy is used, if so whether it is trusted, what is its key management model, what mechanisms it uses to ensure confidentiality, availability, data integrity, as well as any additional security guarantees it provides.

Table 3. Comparison of cryptography in the most representative MCS schemes.

Paper	Protocol	Trusted	Keyless	Confidentiality	Availability	Integrity	Other security properties
[91]	Pond	-	✓	✗	IDA (CRS)	BFT protocols	BFT
[17]	Depsky	-	✓	SKE, PSS	IDA, PSS	Digests of slices	Accountability, BFT
[108]	Hybris	-	✗	SKE (AES)	IDA (RS)	Digests of slices	BFT
[112]	Tahoe	-	✗	SKE (AES)	IDA (RS)	Merkle Trees, DSA	Access Control
[121]	ABE-DSC	-	✗	CP-ABE	✗	✗	Access Control
[3]	RACS	✓	✓	✗	RAID or IDA (RS)	✗	-
[23]	HAIL	-	✓	✗	RAID	PDP, POR	Retrievability
[90]	AONT-RS	-	✓	AONT (AONTRP)	IDA (RS)	Canary	-
[28]	ICStore	-	✓	SKE, PSS	Replication or RAID	DSA on roots of hash trees	BFT
[50]	SecReS	✓	✓	CSKE, PSS	IDA (RS)	Merkle Trees	Access Control
[103]	SDSMC	-	✗	SKE (AES)	Replication	Index-based slices	-
[117]	Gecko	-	✗	CSKE, PSS (LASS)	IDA (RS)	Blockchain (NAP-check)	Accountability, Retrievability
[2]	POMS	✗	✓	SKE, CP-ABE	POR-encoded shares	IBS, POR	Access Control, Auditability, Retrievability
[77]	HealthShare	-	✗	KP-ABE, MA-ABE	Replication	DSA	Access Control
[42]	ABESS	✗	✗	CP-ABE, PSS	IDA, Replication	DSA, HMAC	Access Control, Accountability
[78]	SCUD	-	✓	PSS (PRESS)	PSS	DSA (ECDSA)	-
[66]	CDStore	-	✓	CAONT (COAEP)	IDA (RS)	Digests of slices	-
[46]	NCCloud	✓	✓	✗	IDA ($F-MSR$)	Redundancy checks	-
[116]	CloudSeal	✗	✗	SKE, PRE	-	-	Access Control, Broadcast revocation (SSS)
[31]	LTCS	✓	✗	SKE (AES)	IDA (LT)	Integrity check protocol	Auditability
[97]	CloudS	✓	✓	PCE	PPC	Redundancy checks	-
[119]	SWE	✓	✗	LWE hardness	LWE redundancy	Redundancy checks	-
[60]	KAPRE	✗	✓	SKE, HPRE, PKE, PSS	IDA, PSS	auth enc, commitment HPRF	Accountability
[60]	KAME	✗	✓	SKE, HMKE, PKE, PSS	IDA, PSS	auth enc, commitment HPRF	Accountability

4.1 Trusted Proxy

We consider two approaches which seem semantically similar: either the user trusts a third party delegate, the proxy, to perform the Process/Disperse operations during the upload, or the user himself performs these same operations. The goal of an MCS scheme in this case is to provide security against a coalition of $k-1$ cloud providers. Many protocols were proposed in the literature to achieve this goal.

Keyless approaches. They are divided into three important categories: the first one focuses on protocols that are by construction keyless, we remark that they are generally based on only one primitive, the second category involves protocols that distribute the key separately from the ciphertext to give the keyless aspect to the scheme, and the third category discusses schemes that encapsulate and hide the key within the ciphertext.

Keyless by construction. This first part discusses schemes that do not rely on encryption and thus are keyless by construction. We divide this category into two subcategories.

1. The first one involves MCS schemes that use IDAs, mainly focus on availability, and do not guarantee semantic security (weak to no confidentiality), e.g. [91,3,82,23,46,68,110,39,55].

   Representative schemes. The OceanStore prototype, known as Pond [91], was among the first multi-cloud storage systems to combine erasure coding (using CRS codes) with BFT consensus, ensuring durability and consistency even in hostile environments. Subsequently, the Redundant Array of Cloud Storage (RACS) system [3] demonstrated the feasibility of distributing erasure-coded data across multiple providers to mitigate vendor lock-in. In RACS, a trusted proxy splits objects into chunks, adds redundancy via RAID or RS coding, and disperses the resulting shares among providers; multiple cooperating proxies can also be deployed to avoid single points of failure. Building on this concept, the MCS Scalia [82] introduced optimization-driven orchestration, dynamically adapting data placement to meet cost, durability, and availability objectives through a hardware–software integration running on customer premises. The High-Availability and Integrity Layer (HAIL) scheme [23] further extended the focus to integrity, combining replication with proofs of data possession and proofs of retrievability to verify that redundant fragments remain intact across clouds. The protocol CYRUS [39] is a client-operated MCS that chunks data using Rabin's fingerprinting and uses non systematic RS coding. They claim it is implemented as a (t, n) CSS scheme but the guarantees are only true for a limited number of colluding providers (because confidentiality is partial). This method adds adaptive CSP selection, and supports deduplication and multi-client synchronization. The Network-Coded Cloud (NCCloud) [46] improved repair efficiency through functional minimum-storage regenerating ($\mathsf{F-MSR}$) codes, reducing repair bandwidth while maintaining double-fault tolerance via the MDS property. The DDFM scheme [110] introduces a Granular Data Fragmentation Strategy (based on Granular Computing) to split files into variable-sized fragments, which are then dispersed redundantly across clouds. In addition, DDFM integrates system-level access-control protocols (OpenID, OAuth) for authentication and authorization. The InterCloud RAIDer scheme [68] proposed a fully client-side design that employs High-Speed Resilient Coding (HSRC) for redundancy and a lightweight proof of data-possession mechanism derived from deduplication hashes, providing strong availability and moderate confidentiality without relying on a proxy. Finally, Katarzyna et al. [55] propose a fast fragmentation algorithm that combines ideas from secret sharing and information dispersal. This method emphasizes performance: it disperses, permutes and then encodes data, while availability is guaranteed through erasure codes.

   Advantages and limitations. These protocols are typically lightweight and efficient, as they avoid expensive cryptographic operations such as encryption or key management. They achieve high availability and fault tolerance through erasure coding or replication. Such approaches highlight a trend toward optimizing MCS for speed and scalability rather than relying solely on heavy encryption or secret sharing techniques. The main drawback of these schemes is their lack of

semantic confidentiality, since data fragments are only encoded, a subset of providers may still learn partial information. They assume an honest-but-curious or non-colluding model, which weakens protection against insider threats.

2. The second subcategory includes schemes that leverage secret sharing [5,8,9,78,12,80,26,104,102,70] to jointly guarantee confidentiality and availability without relying on encryption keys.

   Representative schemes. Several MCS schemes [5,8,9] directly adopt Shamir's Secret Sharing (SSS) as their core primitive to ensure confidentiality, integrity, and availability. Protocols such as FVSS [12], Belisarius [80], ARCHISTAR [70] and the work of [118], extend basic secret sharing with efficient verifiability, Byzantine fault tolerance, access control, and trust-optimizing allocation respectively. The protocol SCUD [78] achieves information-theoretic confidentiality via PRESS — a recursive variant of SSS that reduces space inefficiency — complemented by ECDSA signatures for integrity and replication for availability. Potshards [102] combines a full threshold SSS with RAID-style striping for availability but is very inefficient. Other efforts attempt to address architectural enhancements. For instance, SAFE [26] performs share generation and renewal inside a Trusted Execution Environment (TEE) to improve security management. However, this approach reintroduces centralized trust in hardware, weakening the decentralization and openness that multi-cloud cryptographic designs typically aim to preserve. Tchernykh et al. [104] further explore the use of PSS by integrating modular secret sharing (Asmuth–Bloom and Mignotte) with Redundant Residue Number Systems (RRNS) for error detection and correction. Their framework combines threshold-based confidentiality with reliability, demonstrating how secret sharing can be augmented with redundancy to enhance integrity and fault tolerance.

   Advantages and limitations. These schemes provide information-theoric confidentiality and high availability. They are also fast on small to medium files. While these systems demonstrate the simplicity of direct secret sharing, they suffer from significant storage overhead which limits scalability for large datasets.

Keyless by separation. These protocols usually rely on a combination of encryption and IDA for guaranteeing the confidentiality and availability of the data, and rely on secret sharing for securing the key, which creates a tradeoff between security, computation and space.

Representative schemes. The SSMS (Secret Sharing Made Short) cryptographic method, also known as HK1 [59] was introduced by Hugo Krawczyk and became a foundational design pattern for secure MCS, also called computational secret sharing (CSS). It combines SKE with PSS and IDAs to balance efficiency and security: data is first encrypted with a symmetric key, the key is secret-shared among the providers, and the ciphertext is erasure-coded. This hybrid approach minimizes the overhead of secret sharing while retaining its confidentiality guarantees. It was later extended to a robust version called RCSS [94] by using commitment schemes. It has been widely adopted in the world of cloud storage. The SSMS protocol was applied by Bessani et al. in DepSky [17] for fault-tolerant file storage, later extended in SCFS [18] as a file system abstraction adding deduplication and auditing, and redesigned for efficiency and retrievability in CHARON [75]. Variants of this design also appear in works like [79] for their privacy management system, which adds hash-based integrity checks and BFT security, and [52] which adds SSS for user authentication tokens. Another extension of the SSMS model is SecSFE [62], which targets medical data by adding a preliminary fragmentation step before

encryption and secret sharing, and incorporates SHA − 1 signatures with TPA-based auditing for integrity verification. In addition, [122] propose a framework to secure cloud computing by integrating Fully Homomorphic Encryption into Depsky's storage architecture. Its adds a parallelized processing dispatcher to perform computations directly on encrypted data without decryption. This approach enhances confidentiality by keeping data encrypted end-to-end, improves integrity through DepSky's Byzantine fault-tolerant design, and maintains high availability by replicating encoded blocks across multiple clouds. Unlike traditional SSMS schemes that generate keys randomly, the protocol Gecko [117] employs Latin-square Autotopism Secret Sharing (LASS) to secret-share a deterministically constructed key derived from an autotopism, providing a higher level of key protection, but without fault tolerance in the secret-sharing layer. To strengthen integrity, Gecko integrates a blockchain-based audit mechanism: a protocol called NAP-check verifies downloaded slices and pinpoints corrupted ones. SecReS [50] extends the SSMS design pattern in the same way by integrating convergent encryption. Integrity and proof of ownership are guaranteed using Merkle hash trees that allow block-level verification and secure deduplication. Cachin et al. proposed ICStore [28], a dependable intercloud storage system designed to provide confidentiality, integrity, reliability, and consistency (CIRC). ICStore encrypts data with SKE, protects the encryption key using SSS, and verifies integrity through DSA on the roots of per-file hash trees. For reliability, it employs a redundancy layer that can use either replication (high cost but simple) or erasure coding similar to RAID-5 as in RACS or HAIL. To tolerate arbitrary faults among providers, ICStore integrates BFT protocols to maintain consistency during updates, ensuring correct operation even if some clouds behave maliciously.

Advantages and limitations. These schemes have been widely deployed in practice and offer significantly better performance than pure secret-sharing–based approaches for large datasets at the cost of relying on computational rather than information-theoretic security. However, they still require a trusted proxy or client to perform key generation, encryption, and dispersal correctly. Moreover, in order to retrieve the data, trusted processors must perform a 2-step reconstruction: they first reconstruct the key from distributed shares, then use it to decrypt the reconstructed ciphertext.

Keyless by masking. This part discusses schemes that hide the key within the ciphertext and then ensure its availability via IDAs. The most common way to do that and ensure confidentiality of the data and the key is by using AONTs.

Representative schemes. One of the earliest and most widely adopted AONT-based MCS systems is AONT-RS, it serves as the core of several industrial solutions (e.g., Astran, Cleversafe). The scheme was first introduced by Resch and Plank [90], who combined an All-Or-Nothing Transform (AONT) with Reed–Solomon (RS) erasure coding to achieve an efficient and secure storage scheme: the AONT provides confidentiality while RS coding ensures availability. Over time, several variants have been proposed to improve different aspects of security and performance. Lahkar et al. [61] proposed AONT-CRS, replacing RS with Cauchy Reed–Solomon (CRS) codes to improve encoding and decoding performance. Inspired by convergent encryption, [65] introduced CAONT-RS, which replaces the random key of the AONT with a deterministic hash of the data so that identical files produce identical encoded fragments, enabling cross-user deduplication—a property absent from random-key schemes. Similarly, CDStore [66] applied CAONT-RS by combining OAEP-based AONT with RS coding under the same convergent principle. To strengthen integrity and recoverability, Chen et al. [37] presented RAONT-RS, which

integrates commitment schemes to detect corruption and support accountability, replacing the earlier canary-based approach. More recently, [120] proposed SAONT-RS, which introduces an improved AONT and couples it with erasure coding to securely and efficiently handle large data. SAONT-RS addresses several limitations of earlier designs. Other variants explore alternative coding layers. For example, the paper [15] proposed AONT-LT, which combines AONT with Luby Transform (LT) codes. As discussed earlier, LT codes offer high flexibility and low encoding cost, making them a strong alternative to RS, especially when the threshold parameter k is large. In 2019, [115] introduced AONT-NZZD, which pairs an AONT (similar to Rivest's but using hashing instead of encryption) with NZZD erasure coding. AONT-NZZD claims higher performance, while addressing several weaknesses of AONT-RS in terms of reliability, confidentiality, and recoverability.
As an alternative to AONT-IDA approaches, the article [97] introduced CloudS, which follows the same protocol semantic by using a primitive that hides the key within the cipher. It introduces Privacy Protecting Codes (PPC), and combines them with a novel Parallel Cyclic Encryption (PCE) scheme. PPC is first applied to disperse data and achieve availability with low storage and computational overhead, and PCE encrypts each chunk with a random key and hides this key inside another chunk, eliminating centralized key management. The combined PCE–PPC scheme achieves similar security to AONT–RS and better performance depending on the threshold parameter k.

Advantages and limitations. AONT-based schemes provide computational confidentiality without requiring explicit key management as the encryption key is embedded within the transformed data. These schemes are efficient, stateless, and well-suited for large-scale storage, which explains their adoption in several industrial systems. However, they lack formal modeling of AONT constructions under rigorous adversarial settings, and do not address an untrusted proxy model.

Key-leveraging protocols. In this part, we present the protocols that assume the presence of a stored secret on the user's or proxy's side. These schemes generally assume that the user or proxy maintains secret information, including encryption keys, persistent hashes, or identity-proving private keys.

1. The first group of protocols assume that the data owner manages secrets like encryption keys or high value small data.

   Representative schemes. Early schemes like [83,123,103,121,19] focused solely on confidentiality, without providing availability guarantees. For instance, the schemes SDSMC [103] and the one in [83] employ simple data chunking with AES encryption, while integrity is ensured through index-based slicing and hashing, respectively. [123] introduces an authentication and confidentiality mechanism based on homomorphic encryption to protect outsourced data against unauthorized access.
   Other protocols combine encryption with IDAs to add availability. For example, Celesti et al. [34] demonstrate how multi-cloud systems can integrate client-side symmetric encryption with data fragmentation and dispersal to achieve both secrecy and availability. Similarly, Hybris [108] and Tahoe-LAFS [112] follow the SSMS design pattern, where the user retains full control of the encryption key. In Tahoe-LAFS, integrity is ensured through Merkle hash trees and RSA digital signatures, providing verifiable storage. Another cloud storage service that follows the SSMS design, but employs LT codes instead of RS codes is LTCS [31]. To ensure integrity, LTCS

incorporates public verifiability mechanisms that enable third-party auditing of stored fragments without requiring their retrieval, balancing reliability, security, and cost-efficiency. Additionally, several frameworks combine block cipher encryption with deduplication to minimize storage overhead, such as DAC [114] and DupLESS [58]. Taking a different approach, Cao et al. [119] introduced the Storage with Errors (SWE) protocol, a secure and efficient dispersal scheme that integrates an IDA with lattice-based cryptography. As a post-quantum MCS system, SWE leverages the hardness of the Learning With Errors (LWE) problem to guarantee that even if all n fragments are compromised, the original data remains unrecoverable without the secret vector s held by the user. Availability is achieved through the redundancy inherent in LWE, while integrity is verified via a lightweight consistency check using $(k+1)$ fragments.

A different class of schemes applies an invertible Discrete Wavelet Transform (DWT) to split files into fragments with varying levels of sensitivity. In [86], the small, high-value fragment is encrypted using AES, while the remaining fragments are masked via XOR with a $\mathsf{SHA}-3$ digest of the sensitive portion. Similarly, [85] stores the most sensitive fragment in a trusted local environment, while the larger, less critical fragments are outsourced to public clouds. Both approaches reduce encryption overhead while ensuring that no single fragment discloses meaningful information, thus achieving confidentiality through selective protection and availability through dispersal. Building on this concept,the paper [69] proposes an enhanced privacy-preserving data-splitting framework that further improves efficiency by reducing the number of generated fragments and storage operations. In their model, sensitive fragments are retained locally at the proxy, while less sensitive data portions are distributed across multiple CSPs.

Advantages and limitations. Secret-leveraging protocols offer straightforward designs, as they rely on conventional encryption. Despite the fact that they provide strong computational confidentiality, this comes at the cost of explicit key management, which reintroduces a single point of failure, undesirable in MCS. In principle, most of these designs could become keyless by dispersing the secret alongside the data using secret sharing, or transform into untrusted-proxy schemes by assuming that the proxy only performs the Disperse operation.

2. Other schemes rely on users holding private identity keys [121,77,100].

   Representative schemes. Zhang et al. [121] employ CP-ABE for confidentiality and fine-grained access control, where data is categorized into different sensitivity levels, and each level is encrypted under a unified access structure. However, this scheme focuses solely on confidentiality and does not address availability. HealthShare [77] is a client-centric MCS designed for secure electronic health record (EHR) sharing, combining KP-ABE and MA-ABE to provide decentralized trust and fine-grained authorization. Integrity is maintained through digital signatures, while availability is ensured via replication. Similarly, [100] introduces a hybrid framework that integrates symmetric encryption (SKE) with Identity-Based Broadcast Encryption (IBBE) for scalable group key management, in which the data encryption key (DEK) is encrypted once under a broadcast key and distributed across multiple clouds, achieving efficient access control and redundancy-based availability.

   Advantages and limitations. These protocols provide strong confidentiality and fine-grained access control through identity- or attribute-based encryption mechanisms. However, they inherently rely on persistent private or master keys, making them difficult to adapt into fully keyless architectures. Such schemes could be extended to untrusted-proxy settings by delegating only the Disperse operation to the proxy.

4.2 Untrusted proxy

This section discusses the presence of an untrusted proxy, which performs the Disperse operation, while the user carries out the Process phase. Considering an untrusted proxy in MCS settings is crucial, as fully trusting the proxy would undermine the very goal of multi-cloud architectures, which is to achieve distributed trust among independent parties. At the same time, employing a proxy remains practical, as it relieves users from the burden of interacting with multiple cloud providers simultaneously — especially when the user's primary goal is to ensure the availability of their data rather than to manage complex cryptographic or communication tasks. In MCS protocols, even when declared "untrusted", the proxy — by definition a mediator — retains a degree of implicit trust, since it must correctly transmit or transform information during the upload phase for the system to function at all.

Based on the state of the art, the proxy is most often modeled as semi-trusted (or honest-but-curious): it correctly follows the protocol but may attempt to infer confidential information from observed data flows. A stronger and less explored model is that of a malicious proxy operating under the covert adversarial model [13], also referred to as malicious with abort, where the proxy may deviate arbitrarily from the protocol as long as such deviation cannot be detected except with negligible probability. These are reasonable assumptions in the real world, as a proxy, often operated by a company, risks its reputation if caught misbehaving.

Key-leveraging protocols. The simplest way to preserve confidentiality against an untrusted proxy — whether semi-trusted or covert — is to retain a local encryption key, as done in key-leveraging protocols relying on a trusted processor. Such schemes can, in principle, be adapted to an untrusted-proxy setting by restricting the proxy's role to data distribution while keeping all key material inaccessible to it.

Representative schemes. In [74], the authors present a model that emphasizes authentication and integrity rather than confidentiality. The system combines RSA-based user authentication, VPN-secured communication, and Proofs of Data Possession (PDP) for integrity verification. A semi-trusted proxy auditor monitors cloud activity and validates integrity proofs but never holds decryption keys. Availability is maintained through RAID-style redundancy, while users retain the credentials and hashes required for verification. The ABESS protocol [42] supports collaborative sharing of healthcare records across multiple clouds, using CP-ABE combined with SSMS to encrypt and disperse data across independent CSPs. Integrity and authenticity are ensured through Digital Signatures (DSA) on complete records and HMACs on individual shares to detect tampering at the cloud level. However, the scheme incurs substantial computational overhead and long waiting times for multi-party sharing operations. Users and health centers must manage CP-ABE master keys and private keys issued by a central or federated key authority, while each client locally stores DSA and ABE keys for encryption and signing. Rayudu et al. [89] integrate symmetric encryption, proxy re-encryption, and erasure coding to achieve confidentiality, availability, and secure data forwarding. Meanwhile, Lin and Tzeng [67] use threshold PRE combined with erasure coding in order to support secure data forwarding. Key material is distributed among independent key servers using SSS, which perform partial decryption collaboratively. In both designs, the proxy is semi-trusted and assumed non-colluding with key servers, and performs re-encryption without decryption. Users maintain long-term key pairs, and decryption is performed cooperatively through distributed key shares. Similarly, Alfassa et al. [6] propose a multi-cloud framework that combines symmetric encryption with Shamir's

secret sharing for key protection and RSA for encrypting key fragments, alongside improved Proxy Re-Encryption (I-PRE) for controlled data sharing. The proxy re-encrypts data between users without accessing plaintext, while clouds store encrypted fragments of both the data and the key. The proxy is also considered honest-but-curious and non-colluding with CSPs. Users retain their own secret keys, making this another clear instance of a key-leveraging design. CloudSeal [116] is an end-to-end secure content storage and delivery framework providing confidentiality and dynamic access control. Data is encrypted under SKE, and a semi-trusted proxy applies PRE to delegate decryption rights without exposing plaintexts. Upon user revocation, new keys are distributed through threshold secret sharing with broadcast revocation. The proxy is semi-trusted—honest in operation but curious about cached or transiting content—and the scheme focuses on scalable confidentiality rather than integrity or availability.

Advantages and limitations. This family of approaches is practical and realistic, particularly in enterprise or regulated contexts where a proxy naturally exists for coordination or auditing. They provide flexible delegation through re-encryption or attribute-based mechanisms. However, they merely shift the trust boundary: while the proxy no longer learns plaintext, the security now depends on protecting the DEK and other long-term secrets held by users. Key management can be expensive over the years as it requires key rotation. Moreover, an encryption key may be lost, making the data unrecoverable. These schemes also remain vulnerable to insider compromise or key leakage, and few provide formal security models for proxy misbehavior or collusion with cloud providers.

Keyless protocols. We define keyless protocols as those in which the user does not retain any long-term secret keys or hashes during the upload phase. However, during the download phase, in the presence of an untrusted proxy, the user must possess some form of transient secret—such as an ephemeral decryption token—to retrieve data securely. Without this minimal secret material, the proxy would always be able to access the plaintext, even if acting only as an honest-but-curious intermediary.

Representative schemes. In the MCS KAPRE (Keyless Archivage with Proxy Re-Encryption) [60], the user encrypts the data with authenticated SKE and protects the encryption key using SSS combined with homomorphic PRE to ensure its confidentiality. The proxy disperses encrypted data chunks of the data using an IDA. Integrity is verified through key-homomorphic PRF commitments, enabling the proxy to detect corrupted shares and hold providers accountable. KAPRE loses confidentiality if the proxy colludes with any provider. For that reason, the authors presented KAME (Keyless Archivage with Multikey Encryption), which extends KAPRE to tolerate collusion between the proxy and up to $(k-2)$ providers. It replaces proxy re-encryption with homomorphic multi-key encryption (MKE), so that each Shamir coefficient is encrypted under all parties' public keys. Decryption then requires partial decryptions from all participants, ensuring that no single entity — including the proxy — can reconstruct key shares alone. The scheme POMS [2] (Proxy Offloading for Multi-cloud) is an honest-but-curious proxy scheme that aims to improve efficiency while preserving confidentiality, integrity, and retrievability. The user encrypts data with a SKE, encrypts the DEK with CP-ABE for fine-grained access control, and generates a searchable ciphertext and a digital identity-based signature. These components are sent to the proxy, which applies a proof-of-retrievability (POR) encoding to generate and distribute storage shares across multiple clouds. The proxy acts as an auditor, verifying retrievability without learning plaintext or keys. While POMS is considered keyless, it

assumes the existence of a trusted authority that issues temporary secret keys to users during the download phase.

Advantages and limitations. These schemes offer strong security guarantees and practical applicability, particularly in scenarios requiring secure delegation and minimal key management. However, they incur significant computational and communication overhead on the user's side. Moreover, the current state of the art in this area remains limited, with only a few concrete constructions available, highlighting the need for further research into efficient and formally verified keyless designs.

5 Conclusion

This SoK has provided the first systematic study of cryptographic approaches for Multi-Cloud Storage. By classifying protocols along two dimensions — trust models and cryptographic primitives — we clarified how confidentiality, integrity, and availability are achieved under different assumptions and design choices.

Each approach offers distinct strengths: secret sharing achieves strong confidentiality at a high storage cost, erasure coding improves efficiency but leaks partial information, and hybrid constructions balance the two at the expense of complexity. Similarly, the use of trusted or untrusted proxies, keyless designs, or local client memory, introduces different trust assumptions, highlighting that no single model suffices for all settings. Our analysis shows that design trade-offs are unavoidable: stronger guarantees often come with higher storage, communication, or computation overheads.

At the same time, many schemes can be extended depending on security requirements. For example, protocols originally designed for trusted proxies can be adapted to untrusted ones, and keyless designs can be reinforced by secret-sharing or embedding keys inside the data. Likewise, access control, auditability, and robustness features can be layered on top of existing storage schemes. Ideally, a dispersed encryption key or sensitive fragment should enjoy stronger protection than the file it secures — a principle guiding future hybrid designs.

In summary, our systematization reveals that MCS security is not a solved problem but a rich design space where cryptography, distributed systems, and trust models intersect. We hope this SoK helps unify terminology, identify open challenges, and guide future designs toward secure, efficient, and trustworthy multi-cloud architectures.

We particularly encourage further research in this field, especially in the exploration of keyless and untrusted-proxy models, which remain among the least developed yet most promising directions for future MCS security. Another open avenue concerns the integration of post-quantum cryptography into multi-cloud settings, as exemplified by lattice-based schemes such as SWE, which illustrates the need for quantum-resilient yet efficient and creative integration of post-quantum techniques.

Acknowledgments

We would like to express our gratitude to Nigel Smart for his insightful comments and frequent discussions on the manuscript, and to the Astran Crypto team, specifically Gilles Seghaier, Elouan Gros, and Daoud Jahdou, for their comments, helpful advice, and valuable perspectives. We also thank the anonymous reviewers for their constructive feedback, which helped to improve the quality and clarity of this paper.

This work was supported by the joint research laboratory of EPITA and Astran, and by the France 2030 innovation program of Bpifrance and the French government.

References

1. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In: 4th USENIX Conference on File and Storage Technologies (FAST 05). USENIX Association, San Francisco, CA (Dec 2005), https://www.usenix.org/conference/fast-05/weaver-codes-highly-fault-tolerant-erasure-codes-storage-systems
2. Abdel-Rahman, A.O., Azogagh, S., Birba, Z.A., Van, A.T.: POMS : Proxy offloading for multicloud storage with keyword search. Cryptology ePrint Archive, Paper 2024/1747 (2024), https://eprint.iacr.org/2024/1747
3. Abu-Libdeh, H., Princehouse, L., Weatherspoon, H.: Racs: a case for cloud storage diversity. In: Proceedings of the 1st ACM Symposium on Cloud Computing. p. 229–240. SoCC '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1807128.1807165, https://doi.org/10.1145/1807128.1807165
4. Adei, D., Orsini, C., Scafuro, A., Verber, T.: How to recover a cryptographic secret from the cloud. Cryptology ePrint Archive, Paper 2023/1308 (2023), https://eprint.iacr.org/2023/1308
5. Alam, M., Kather, S.: An approach to secret sharing algorithm in cloud computing security over single to multi clouds. International Journal of Scientific and Research Publications **3** (04 2013)
6. Alfassa, S.M., Padmaveni, K.: Improved availability using i_rrect algorithm in cloud environment. In: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). pp. 625–630 (2018). https://doi.org/10.1109/ICCONS.2018.8663034
7. Alqahtani, H.: A Novel Approach to Providing Secure Data Storage Using Multi Cloud Computing. University of Bedfordshire (2019), https://books.google.fr/books?id=Mq-TzgEACAAJ
8. Alsolami, F., Boult, T.E.: Cloudstash: Using secret-sharing scheme to secure data, not keys, in multi-clouds. In: 2014 11th International Conference on Information Technology: New Generations. pp. 315–320 (2014). https://doi.org/10.1109/ITNG.2014.119
9. Althamary, I., Alkharobi, T.: Secure file sharing in multi-clouds using shamir's secret sharing scheme. Transactions on Networks and Communications **4** (12 2016). https://doi.org/10.14738/tnc.46.2560
10. AlZain, M.A., Pardede, E., Soh, B., Thom, J.A.: Cloud computing security: From single to multi-clouds. In: 2012 45th Hawaii International Conference on System Sciences. pp. 5490–5499 (2012). https://doi.org/10.1109/HICSS.2012.153
11. Attasena, V., Darmont, J., Harbi, N.: Secret sharing for cloud data security: a survey. The VLDB Journal **26**(5), 657–681 (Oct 2017). https://doi.org/10.1007/s00778-017-0470-9, https://doi.org/10.1007/s00778-017-0470-9
12. Attasena, V., Harbi, N., Darmont, J.: fVSS: A New Secure and Cost-Efficient Scheme for Cloud Data Warehouses. In: ACM (ed.) Proceedings of the 17th International Workshop on Data Warehousing and OLAP (DOLAP 2014), Shangai, China. pp. 81–90. Proceedings of the 17th International Workshop on Data Warehousing and OLAP (DOLAP 2014), Shangai, China, Shangai, China (Nov 2014). https://doi.org/10.1145/2666158.2666173, https://hal.science/hal-01081882
13. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. Cryptology ePrint Archive, Paper 2007/060 (2007), https://eprint.iacr.org/2007/060
14. Balaji, S.B., Krishnan, M.N., Vajha, M., Ramkumar, V., Sasidharan, B., Kumar, P.V.: Erasure coding for distributed storage: An overview (2018), https://arxiv.org/abs/1806.04437
15. Baldi, M., Maturo, N., Montali, E., Chiaraluce, F.: Aont-lt: A data protection scheme for cloud and cooperative storage systems. In: 2014 International Conference on High Performance Computing & Simulation (HPCS). pp. 566–571 (2014). https://doi.org/10.1109/HPCSim.2014.6903736

16. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A.D. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (May 1995). https://doi.org/10.1007/BFb0053428
17. Bessani, A., Correia, M., Quaresma, B., André, F., Sousa, P.: Depsky: Dependable and secure storage in a cloud-of-clouds. ACM Trans. Storage **9**(4) (nov 2013). https://doi.org/10.1145/2535929, https://doi.org/10.1145/2535929
18. Bessani, A., Mendes, R., Oliveira, T., Neves, N., Correia, M., Pasin, M., Veríssimo, P.: Scfs: A shared cloud-backed file system. pp. 169–180 (01 2014)
19. Bharot, N., Mehta, N., Breslin, J., Verma, P.: Cloudlock: secure data sharing using a hybrid cryptosystem in multi-cloud data storage. Cluster Computing **28** (07 2025). https://doi.org/10.1007/s10586-025-05433-7
20. Blakley, G.R., Meadows, C.: Security of ramp schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 242–268. Springer, Heidelberg (Aug 1984). https://doi.org/10.1007/3-540-39568-7_20
21. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. vol. 1403, pp. 127–144 (05 1998). https://doi.org/10.1007/BFb0054122
22. Bohli, J.M., Gruschka, N., Jensen, M., Iacono, L.L., Marnau, N.: Security and privacy-enhancing multicloud architectures. IEEE Transactions on Dependable and Secure Computing **10**(4), 212–224 (2013). https://doi.org/10.1109/TDSC.2013.6
23. Bowers, K.D., Juels, A., Oprea, A.: HAIL: A high-availability and integrity layer for cloud storage. Cryptology ePrint Archive, Paper 2008/489 (2008), https://eprint.iacr.org/2008/489
24. Boyd, C., Davies, G.T., Gjøsteen, K., Toorani, M., Raddum, H.: Security notions for cloud storage and deduplication. Cryptology ePrint Archive, Paper 2017/1208 (2017), https://eprint.iacr.org/2017/1208
25. Boyko, V.: On the security properties of OAEP as an all-or-nothing transform. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 503–518. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_32
26. Buchmann, J., Dessouky, G., Frassetto, T., Ágnes Kiss, Sadeghi, A.R., Schneider, T., Traverso, G., Zeitouni, S.: SAFE: A secure and efficient long-term distributed storage system. Cryptology ePrint Archive, Paper 2020/690 (2020), https://eprint.iacr.org/2020/690
27. Buyya, R., Ranjan, R., Calheiros, R.: Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. vol. 6081 (03 2010). https://doi.org/10.1007/978-3-642-13119-6_2
28. Cachin, C., Haas, R., Vukolic, M.: Dependable storage in the intercloud. IBM Research **3783**, 1–6 (01 2010)
29. Cachin, C., Keidar, I., Shraer, A.: Trusting the cloud. SIGACT News **40**(2), 81–86 (Jun 2009). https://doi.org/10.1145/1556154.1556173, https://doi.org/10.1145/1556154.1556173
30. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_33
31. Cao, N., Yu, S., Yang, Z., Lou, W., Hou, Y.T.: Lt codes-based secure and reliable cloud storage service. In: 2012 Proceedings IEEE INFOCOM. pp. 693–701 (2012). https://doi.org/10.1109/INFCOM.2012.6195814
32. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. p. 173–186. OSDI '99, USENIX Association, USA (1999)
33. Celesti, A., Galletta, A., Fazio, M., Villari, M.: Towards hybrid multi-cloud storage systems: Understanding how to perform data transfer. Big Data Research **16**, 1–17 (2019). https://doi.org/https://doi.org/10.1016/j.bdr.2019.02.002, https://www.sciencedirect.com/science/article/pii/S2214579618302004
34. Celesti, A., Tusa, F., Villari, M., Puliafito, A.: Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. Journal of Network and Computer Applications **34**(4), 1216–1227 (2011). https://doi.org/10.1016/j.jnca.2010.05.009

35. Chattopadhyay, A.K., Saha, S., Nag, A., Nandi, S.: Secret sharing: A comprehensive survey, taxonomy and applications. Computer Science Review **51**, 100608 (2024). https://doi.org/https://doi.org/10.1016/j.cosrev.2023.100608, https://www.sciencedirect.com/science/article/pii/S1574013723000758
36. Chen, H., Cramer, R.: Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (Aug 2006). https://doi.org/10.1007/11818175_31
37. Chen, L., Laing, T.M., Martin, K.M.: Revisiting and extending the AONT-RS scheme: A robust computationally secure secret sharing scheme. In: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 17. LNCS, vol. 10239, pp. 40–57. Springer, Heidelberg (May 2017). https://doi.org/10.1007/978-3-319-57339-7_3
38. Chockler, G., Guerraoui, R., Keidar, I., Vukolic, M.: Reliable distributed storage. Computer **42** (05 2009). https://doi.org/10.1109/MC.2009.126
39. Chung, J.Y., Joe-Wong, C., Ha, S., Hong, J.W.K., Chiang, M.: Cyrus: towards client-defined cloud storage. In: Proceedings of the Tenth European Conference on Computer Systems. EuroSys '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2741948.2741951, https://doi.org/10.1145/2741948.2741951
40. Dimakis, A.G., Godfrey, P.B., Wu, Y., Wainwright, M.J., Ramchandran, K.: Network coding for distributed storage systems. IEEE Transactions on Information Theory **56**(9), 4539–4551 (2010). https://doi.org/10.1109/TIT.2010.2054295
41. Esfahani, A.: Generalizations of All-or-Nothing Transforms and their Application in Secure Distributed Storage. Phd thesis, University of Waterloo, Waterloo (2021)
42. Fabian, B., Ermakova, T., Junghanns, P.: Collaborative and secure sharing of healthcare data in multi-clouds. Information Systems **48**, 132–150 (05 2014). https://doi.org/10.1016/j.is.2014.05.004
43. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009), crypto.stanford.edu/craig
44. Hong, J., Dreibholz, T., Schenkel, J., Hu, J.: An Overview of Multi-cloud Computing, pp. 1055–1068 (03 2019). https://doi.org/10.1007/978-3-030-15035-8_103
45. Hu, P., Sung, C.W., Ho, S.W., Chan, T.H.: Optimal coding and allocation for perfect secrecy in multiple clouds. IEEE Transactions on Information Forensics and Security **11**(2), 388–399 (2016). https://doi.org/10.1109/TIFS.2015.2500193
46. Hu, Y., Chen, H.C.H., Lee, P.P.C., Tang, Y.: Nccloud: applying network coding for the storage repair in a cloud-of-clouds. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies. p. 21. FAST'12, USENIX Association, USA (2012)
47. Hussein, N., Salih, A., Khanfar, K.: A survey of cryptography cloud storage techniques. International Journal of Computer Science and Mobile Computing **52**, 186–191 (03 2016)
48. ImmuniWeb: Top 10 cloud security incidents in 2022 (2022), https://www.immuniweb.com/blog/top-10-cloud-security-incidents-in-2022.html
49. Imran, H.A., Latif, U., Ikram, A.A., Ehsan, M., Ikram, A.J., Khan, W.A., Wazir, S.: Multi-cloud: A comprehensive review. In: 2020 IEEE 23rd International Multitopic Conference (INMIC). pp. 1–5 (2020). https://doi.org/10.1109/INMIC50486.2020.9318176
50. Islam, T., Mistareehi, H., Manivannan, D.: Secres: A secure and reliable storage scheme for cloud with client-side data deduplication. In: 2019 IEEE Global Communications Conference (GLOBECOM). pp. 1–6 (2019). https://doi.org/10.1109/GLOBECOM38437.2019.9013469
51. Kaaniche, N., Laurent, M.: Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms. Computer Communications **111** (08 2017). https://doi.org/10.1016/j.comcom.2017.07.006
52. Kadam, M., Chaudhary, S., Carvalho, B.: Security approach for multi-cloud data storage. International Journal of Computer Applications **126**, 27–31 (09 2015). https://doi.org/10.5120/ijca2015906032
53. Kannan, M., Naik, H.: A survey on protecting confidential data over distributed storage in cloud. SSRN Electronic Journal **5**, 1–7 (01 2021). https://doi.org/10.2139/ssrn.3740465

54. Kapusta, K.: Protecting data confidentiality combining data fragmentation, encryption, and dispersal over a distributed environment. Phd thesis, Télécom ParisTech, Paris, France (2018), english. NNT: 2018ENST0061. tel-03523671
55. Kapusta, K., Memmi, G.: A fast fragmentation algorithm for data protection in a multi-cloud environment (2018), https://arxiv.org/abs/1804.01886
56. Karame, G., Soriente, C., Lichota, K., Capkun, S.: Securing cloud data under key exposure. IEEE Transactions on Cloud Computing **PP**, 1–1 (02 2017). https://doi.org/10.1109/TCC.2017.2670559
57. Kathuria, S.: A survey on security provided by multi-clouds in cloud computing. International Journal of Scientific Research in Network Security and Communication **6**(1), 23–27 (2018). https://doi.org/10.26438/ijsrnsc/v6i1.2327, https://ijsrnsc.org/index.php/j/article/view/120/120
58. Keelveedhi, S., Bellare, M., Ristenpart, T.: DupLESS: Server-Aided encryption for deduplicated storage. In: 22nd USENIX Security Symposium (USENIX Security 13). pp. 179–194. USENIX Association, Washington, D.C. (Aug 2013), https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/bellare
59. Krawczyk, H.: Secret sharing made short. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 136–146. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_12
60. Lafourcade, P., Mallordy, L.B., Olivier-Anclin, C., Robert, L.: Secure Keyless Multi-Party Storage Scheme. In: ESORICS : European Symposium On Research In Computer Security. Lecture Notes in Computer Science, vol. LNCS - 14984, pp. 279–298. Springer Nature Switzerland, Bydgoszcz, Poland (Sep 2024). https://doi.org/10.1007/978-3-031-70896-1_14, https://hal.science/hal-04540895
61. Lahkar, H., R., M.C.: Towards high security and fault tolerant dispersed storage system with optimized information dispersal algorithm. International Journal of Advanced Research in Computer Science **5**(6), 167–172 (2014), https://www.ijarcs.info/index.php/Ijarcs/article/view/2205
62. Le, D.N., Seth, B., Dalal, S.: A hybrid approach of secret sharing with fragmentation and encryption in cloud environment for securing outsourced medical database: A revolutionary approach. Journal of Cyber Security and Mobility **7**(4), 379–408 (Mar 2018). https://doi.org/10.13052/2245-1439.742, https://journals.riverpublishers.com/index.php/JCSANDM/article/view/5311
63. Li, J., Yan, H., Zhang, Y.: Efficient identity-based provable multi-copy data possession in multi-cloud storage. IEEE Transactions on Cloud Computing **10**(1), 356–365 (2022). https://doi.org/10.1109/TCC.2019.2929045
64. Li, J., Krohn, M., Mazières, D., Shasha, D.: Secure untrusted data repository (SUNDR). In: 6th Symposium on Operating Systems Design & Implementation (OSDI 04). USENIX Association, San Francisco, CA (Dec 2004), https://www.usenix.org/conference/osdi-04/secure-untrusted-data-repository-sundr
65. Li, M., Qin, C., Lee, P.P.C., Li, J.: Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds. In: Kozuch, M.A., Yu, M. (eds.) 6th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud '14, Philadelphia, PA, USA, June 17-18, 2014. USENIX Association (2014), https://www.usenix.org/conference/hotstorage14/workshop-program/presentation/li_mingqiang
66. Li, M., Qin, C., Li, J., Lee, P.: Cdstore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. IEEE Internet Computing **20** (02 2016). https://doi.org/10.1109/MIC.2016.45
67. Lin, H.Y., Tzeng, W.G.: A secure erasure code-based cloud storage system with secure data forwarding. IEEE Trans. Parallel Distrib. Syst. **23**(6), 995–1003 (Jun 2012). https://doi.org/10.1109/TPDS.2011.252, https://doi.org/10.1109/TPDS.2011.252
68. Ling, C., Datta, A.: Intercloud raider: A do-it-yourself multi-cloud private data backup system. vol. 8314, pp. 453–468 (01 2014). https://doi.org/10.1007/978-3-642-45249-9_30
69. Loh, R., Thing, V.L.L.: Data privacy in multi-cloud: An enhanced data fragmentation framework. In: 2021 18th International Conference on Privacy, Security and Trust (PST). pp. 1–5 (2021). https://doi.org/10.1109/PST52912.2021.9647746

70. Loruenser, T., Happe, A., Slamanig, D.: Archistar: Towards secure and robust cloud based data sharing. In: 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom). pp. 371–378 (2015). https://doi.org/10.1109/CloudCom.2015.71
71. Lu, Y., Li, J.: A pairing-free certificate-based proxy re-encryption scheme for secure data sharing in public clouds. Future Gener. Comput. Syst. **62**(C), 140–147 (Sep 2016). https://doi.org/10.1016/j.future.2015.11.012, https://doi.org/10.1016/j.future.2015.11.012
72. Malkhi, D., Reiter, M.: Byzantine quorum systems. Distrib. Comput. **11**(4), 203–213 (Oct 1998). https://doi.org/10.1007/s004460050050, https://doi.org/10.1007/s004460050050
73. Mazières, D., Shasha, D.: Building secure file systems out of byzantine storage. In: Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing. p. 108–117. PODC '02, Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/571825.571840, https://doi.org/10.1145/571825.571840
74. Megouache, L., Zitouni, A., Djoudi, M.: Ensuring user authentication and data integrity in multi-cloud environment. Human-centric Computing and Information Sciences **10** (12 2020). https://doi.org/10.1186/s13673-020-00224-y
75. Mendes, R., Oliveira, T., Cogo, V., Neves, N., Bessani, A.: Charon: A secure cloud-of-clouds system for storing and sharing big data. IEEE Transactions on Cloud Computing **9**(4), 1349–1361 (2021). https://doi.org/10.1109/TCC.2019.2916856
76. Miao, Y., Huang, Q., Xiao, M., Susilo, W.: Blockchain assisted multi-copy provable data possession with faults localization in multi-cloud storage. IEEE Transactions on Information Forensics and Security **17**, 3663–3676 (2022). https://doi.org/10.1109/TIFS.2022.3211642
77. Michalas, A., Weingarten, N.: Healthshare: Using attribute-based encryption for secure data sharing between multiple clouds. In: 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS). pp. 811–815 (2017). https://doi.org/10.1109/CBMS.2017.30
78. Niknia, A., Correia, M., Karimpour, J.: Secure cloud-of-clouds storage with space-efficient secret sharing. Cryptology ePrint Archive, Paper 2021/666 (2021), https://eprint.iacr.org/2021/666
79. Pachala, S., Rupa, C., Lingamgunta, S.: An improved security and privacy management system for data in multi-cloud environments using a hybrid approach. Evolutionary Intelligence **14** (06 2021). https://doi.org/10.1007/s12065-020-00555-w
80. Padilha, R., Pedone, F.: Belisarius: Bft storage with confidentiality. In: Proceedings of the 2011 IEEE 10th International Symposium on Network Computing and Applications. p. 9–16. NCA '11, IEEE Computer Society, USA (2011). https://doi.org/10.1109/NCA.2011.15, https://doi.org/10.1109/NCA.2011.15
81. Pal, D., Khethavath, P., Thomas, J.P., Chen, T.: Multilevel threshold secret sharing in distributed cloud. In: Abawajy, J.H., Mukherjea, S., Thampi, S.M., Ruiz-Martínez, A. (eds.) Security in Computing and Communications. pp. 13–23. Springer International Publishing, Cham (2015)
82. Papaioannou, T.G., Bonvin, N., Aberer, K.: Scalia: An adaptive scheme for efficient multi-cloud storage. In: SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 1–10 (2012). https://doi.org/10.1109/SC.2012.101
83. Pathak, N.N., Nagori, M.: Enhanced security for multi cloud storage using aes algorithm (2015), https://api.semanticscholar.org/CorpusID:203689128
84. Plank, J.S.: Erasure codes for storage systems (2013), https://api.semanticscholar.org/CorpusID:3238809
85. Qiu, H., Memmi, G., Noura, H.: An efficient secure storage scheme based on information fragmentation. In: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). pp. 108–113 (2017). https://doi.org/10.1109/CSCloud.2017.44
86. Qiu, H., Noura, H., Qiu, M., Ming, Z., Memmi, G.: A user-centric data protection method for cloud storage based on invertible dwt. IEEE Transactions on Cloud Computing **9**(4), 1293–1304 (2021). https://doi.org/10.1109/TCC.2019.2911679
87. Quisquater, M., Preneel, B., Vandewalle, J.: On the security of the threshold scheme based on the chinese remainder theorem. In: Proceedings of the 5th International Workshop on Practice and Theory

in Public Key Cryptosystems: Public Key Cryptography. p. 199–210. PKC '02, Springer-Verlag, Berlin, Heidelberg (2002)
88. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. J. ACM **36**(2), 335–348 (1989). https://doi.org/10.1145/62044.62050, https://doi.org/10.1145/62044.62050
89. Rayudu, V.D., Vazralu, M., Sandeep, M.: A secure erasure code-based cloud storage system with secure data forwarding. International Journal of Computer Science and Mobile Computing (IJCSMC) **3**(8), 337–344 (August 2014), https://www.ijcsmc.com/docs/papers/August2014/V3I8201422.pdf
90. Resch, J.K., Plank, J.S.: AONT-RS: blending security and performance in dispersed storage systems. In: Ganger, G.R., Wilkes, J. (eds.) 9th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, February 15-17, 2011. pp. 191–202. USENIX (2011), http://www.usenix.org/events/fast11/tech/techAbstracts.html#Resch
91. Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., Kubiatowicz, J.: Pond: The OceanStore prototype. In: 2nd USENIX Conference on File and Storage Technologies (FAST 03). USENIX Association, San Francisco, CA (Mar 2003), https://www.usenix.org/conference/fast-03/pond-oceanstore-prototype
92. Rivest, R.L.: All-or-nothing encryption and the package transform. In: Biham, E. (ed.) FSE'97. LNCS, vol. 1267, pp. 210–218. Springer, Heidelberg (Jan 1997). https://doi.org/10.1007/BFb0052348
93. Rocha, F.L., Correia, M.P.: Lucy in the sky without diamonds: Stealing confidential data in the cloud. 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W) pp. 129–134 (2011), https://api.semanticscholar.org/CorpusID:11517338
94. Rogaway, P., Bellare, M.: Robust computational secret sharing and a unified account of classical secret-sharing goals. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007. pp. 172–184. ACM Press (Oct 2007). https://doi.org/10.1145/1315245.1315268
95. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176
96. Shen, J., Shen, J., Chen, X., Huang, X., Susilo, W.: An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Transactions on Information Forensics and Security **12**(10), 2402–2415 (2017). https://doi.org/10.1109/TIFS.2017.2705620
97. Shen, L., Feng, S., Sun, J., Li, Z., Wang, G., Liu, X.: Clouds: A multi-cloud storage system with multi-level security. In: Wang, G., Zomaya, A., Martinez, G., Li, K. (eds.) Algorithms and Architectures for Parallel Processing. pp. 703–716. Springer International Publishing, Cham (2015)
98. Shor, R., Yadgar, G., Huang, W., Yaakobi, E., Bruck, J.: How to best share a big secret. In: Proceedings of the 11th ACM International Systems and Storage Conference. p. 76–88. SYSTOR '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3211890.3211896, https://doi.org/10.1145/3211890.3211896
99. Smart, N.P.: Cryptography Made Simple. ISC, Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-21936-3
100. Sohal, M., Bharany, S., Sharma, S., Maashi, M.S., Aljebreen, M.: A hybrid multi-cloud framework using the ibbe key management system for securing data storage. Sustainability **14**(20) (2022). https://doi.org/10.3390/su142013561, https://www.mdpi.com/2071-1050/14/20/13561
101. Stinson, D.: Something about all or nothing (transforms). Designs Codes and Cryptography **22**, 133–138 (01 2001). https://doi.org/10.1023/A:1008304703074
102. Storer, M.W., Greenan, K.M., Miller, E.L., Voruganti, K.: Potshards—a secure, recoverable, long-term archival storage system. ACM Trans. Storage **5**(2) (Jun 2009). https://doi.org/10.1145/1534912.1534914, https://doi.org/10.1145/1534912.1534914
103. Subramanian, K., John, F.: Enhanced security for data sharing in multi cloud storage (sdsmc). International Journal of Advanced Computer Science and Applications **8** (2017), https://api.semanticscholar.org/CorpusID:5675637
104. Tchernykh, A., Miranda, V., Babenko, M., Armenta, F., Radchenko, G., Drozdov, A., Avetisyan, A.: Performance evaluation of secret sharing schemes with data recovery in secured and reliable

heterogeneous multi-cloud storage. Cluster Computing **22** (12 2019). https://doi.org/10.1007/s10586-018-02896-9
105. Thilakanathan, D., Chen, S., Nepal, S., Calvo, R.: Secure Data Sharing in the Cloud (07 2014). https://doi.org/10.1007/978-3-642-38586-5_2
106. Ukwandu, E.: RESCUE: evaluation of a fragmented secret share system in distributed-cloud architecture. Ph.D. thesis, Edinburgh Napier University, UK (2019)
107. Vaishali Tupe, V.D.: Data security in multi-cloud environment a survey. International Conference and Workshop on Communication, Computing and Virtualization **ICWCCV2015**(1), 0–0 (September 2015), /proceedings/icwccv2015/number1/785-1552/
108. Viotti, P., Dobre, D., Vukolić, M.: Hybris: Robust hybrid cloud storage. ACM Trans. Storage **13**(3) (Sep 2017). https://doi.org/10.1145/3119896, https://doi.org/10.1145/3119896
109. Vukolić, M.: The byzantine empire in the intercloud. SIGACT News **41**(3), 105–111 (Sep 2010). https://doi.org/10.1145/1855118.1855137, https://doi.org/10.1145/1855118.1855137
110. Wang, L., Liu, L., Liu, S., Chen, D., Chen, Y.: A secured distributed and data fragmentation model for cloud storage. Applied Mechanics and Materials **347-350**, 2693–2699 (08 2013). https://doi.org/10.4028/www.scientific.net/AMM.347-350.2693
111. Weatherspoon, H., Kubiatowicz, J.D.: Erasure coding vs. replication: A quantitative comparison. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) Peer-to-Peer Systems. pp. 328–337. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
112. Wilcox-O'Hearn, Z., Warner, B.: Tahoe – the least-authority filesystem. Cryptology ePrint Archive, Paper 2012/524 (2012), https://eprint.iacr.org/2012/524
113. Witanto, E.N., Stanley, B., Lee, S.G.: Distributed data integrity verification scheme in multi-cloud environment. Sensors **23**(3) (2023). https://doi.org/10.3390/s23031623, https://www.mdpi.com/1424-8220/23/3/1623
114. Wu, S., Li, K.C., Mao, B., Liao, M.: Dac: Improving storage availability with deduplication-assisted cloud-of-clouds. Future Generation Computer Systems **74**, 190–198 (2017). https://doi.org/https://doi.org/10.1016/j.future.2016.02.001, https://www.sciencedirect.com/science/article/pii/S0167739X16300073
115. Xie, P., Li, H., Yu, H., Chen, Z.: AONT-NZZD: A Secure and Efficient Dispersal Scheme in Distributed Storage Systems . In: 2019 IEEE International Conference on Big Data (Big Data). pp. 6232–6239. IEEE Computer Society, Los Alamitos, CA, USA (Dec 2019). https://doi.org/10.1109/BigData47090.2019.9005571, https://doi.ieeecomputersociety.org/10.1109/BigData47090.2019.9005571
116. Xiong, H., Zhang, X., Zhu, W., Yao, D.: Cloudseal: End-to-end content protection in cloud-based storage and delivery services. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) Security and Privacy in Communication Networks. pp. 491–500. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
117. Yan, M., Feng, J., Marbach, T., Stones, R., Wang, G., Liu, X.: Gecko: A resilient dispersal scheme for multi-cloud storage. IEEE Access **PP**, 1–1 (06 2019). https://doi.org/10.1109/ACCESS.2019.2920405
118. Yang, J., Zhu, H., Liu, T.: Secure and economical multi-cloud storage policy with nsga-ii-c. Applied Soft Computing **83**, 105649 (2019). https://doi.org/https://doi.org/10.1016/j.asoc.2019.105649, https://www.sciencedirect.com/science/article/pii/S1568494619304296
119. Yang, L., Fang, F., Lu, X., Zhu, W.T., Wang, Q., Yan, S., Pan, S.: A secure and fast dispersal storage scheme based on the learning with errors problem. In: Security and Privacy in Communication Networks (SecureComm 2016). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 198, pp. 392–411. Springer (2016). https://doi.org/10.1007/978-3-319-59608-2_23, https://doi.org/10.1007/978-3-319-59608-2_23
120. Yao, L., Lu, J., Liu, J., Wang, D., Meng, B.: A secure and efficient distributed storage scheme saont-rs based on an improved aont and erasure coding. IEEE Access **6**, 55126–55138 (2018). https://doi.org/10.1109/ACCESS.2018.2872749

121. Zhang, W., Zhang, Y., Shi, Q., Liu, Q., Liao, X.: An efficient attribute-based encryption scheme with data security classification in the multi-cloud environment. Electronics **12**(20), 4237 (2023). https://doi.org/10.3390/electronics12204237, https://www.mdpi.com/2079-9292/12/20/4237
122. Zibouh, O., Dalli, A., Drissi, H.: Cloud computing security through parallelizing fully homomorphic encryption applied to multi-cloud approach (2016), https://api.semanticscholar.org/CorpusID:27115275
123. Zkik, K., Orhanou, G., Hajji, S.: Secure mobile multi cloud architecture for authentication and data storage. International Journal of Cloud Applications and Computing (IJCAC) **7**, 62–76 (04 2017). https://doi.org/10.4018/IJCAC.2017040105