



A Secure Cloud-Based Framework for Distributed Data Storage and Access Control

Liron Qudratullah

Department of Marine and Offshore Engineering, Borneo School of Business and Technology, Malaysia
liron.qudratullah@bsbt-my.org

Peer Review Information	Abstract
<i>Submission: 18 Jan 2019</i> <i>Revision: 02 Feb 2019</i> <i>Acceptance: 18 March 2019</i> Keywords <i>Cloud Security, Distributed Storage, Attribute-Based Encryption, Access Control, Data Integrity</i>	Cloud storage offers elastic capacity and convenient data sharing, but outsourcing sensitive information to remote infrastructure introduces risks related to confidentiality, integrity, availability, and authorization. This methodology paper proposes a secure cloud-based framework that combines client-side symmetric encryption, ciphertext-policy attribute-based encryption (CP-ABE), distributed storage, integrity verification, and revocation-aware access control. Data are encrypted with AES-256-GCM before leaving the client, while the symmetric data key is protected by a CP-ABE policy so that decryption depends on user attributes rather than a single account identity. Encrypted blocks are distributed across independent storage nodes with replication, and cryptographic hashes together with proof-of-possession/retrievability concepts are used to support integrity checking. The methodology defines system entities, threat assumptions, upload and access workflows, revocation procedures, auditing, and a prototype evaluation procedure. A small local benchmark shows near-linear cryptographic processing as file size increases, while functional tests demonstrate that unauthorized attributes, modified ciphertext, and revoked credentials are rejected. The framework therefore provides a practical layered design for secure distributed cloud storage without requiring the cloud provider to learn plaintext data or hold unrestricted decryption keys.

Introduction

Cloud computing has made remote data storage a standard component of modern information systems because it reduces local infrastructure cost, supports elastic capacity, and enables users to access shared resources from multiple locations. The same outsourcing model, however, changes the security boundary. Once a file is placed on infrastructure controlled by a third party, the data owner must assume that storage servers can fail, administrators may be curious, credentials can be stolen, and network communication can be intercepted. Conventional server-side access control is not sufficient when the server itself is outside the owner's full trust domain. A secure storage design should therefore

protect data cryptographically before upload and should maintain security even if one storage component is compromised.

Fine-grained cryptographic access control is particularly suitable for shared cloud environments. Bethencourt, Sahai, and Waters (2007) introduced ciphertext-policy attribute-based encryption (CP-ABE), in which a ciphertext carries an access policy and a user can decrypt only when the attributes in the user's secret key satisfy that policy. This is more expressive than distributing a separate encryption key to every individual user. Yu et al. (2010) later applied attribute-based techniques to cloud computing and demonstrated how fine-grained policies can

be enforced while data remain encrypted. These ideas motivate an architecture in which the cloud stores ciphertext but does not independently decide who can read the underlying plaintext.

Confidentiality alone is insufficient for distributed storage. A cloud node could lose, truncate, or modify outsourced blocks while still appearing available. Ateniese et al. (2007) introduced provable data possession (PDP), while Juels and Kaliski (2007) formalized proofs of retrievability (POR). Both approaches allow a verifier to check remote data possession or retrievability without downloading the complete file. Wang et al. (2010) further considered privacy-preserving public auditing for cloud storage. These mechanisms indicate that a practical framework should treat access control and integrity verification as complementary services rather than isolated functions.

The objective of this paper is to define a methodology for a secure cloud-based framework for distributed data storage and access control. The proposed design uses client-side AES-256-GCM for efficient bulk encryption, CP-ABE for protecting data-encryption keys, distributed replicated storage for resilience, and integrity metadata for remote verification. The methodology also includes attribute revocation, authentication, access logging, and a least-trust separation between the cloud provider and the key authority. The work is presented as a reproducible design and prototype evaluation method rather than as a claim of production-scale deployment.

Literature Review

Bethencourt et al. (2007) established CP-ABE as a mechanism for enforcing expressive access policies over encrypted data. Instead of encrypting independently for every recipient, an owner can define a logical rule such as "Department = Research AND Role = Faculty." Only users whose certified attributes satisfy that rule can recover the protected secret. CP-ABE is therefore a natural building block for cloud data sharing because policy enforcement is tied to cryptography rather than to the honesty of the storage server. Its main practical limitations are computational overhead, key management, and revocation when attributes change.

Remote integrity verification was developed in parallel with access-control research. Ateniese et al. (2007) proposed PDP so that a client can probabilistically verify that an untrusted server still possesses outsourced data using compact challenge-response proofs. Juels and Kaliski (2007) proposed POR to give stronger assurance

that a complete file remains retrievable. Wang et al. (2010) adapted public auditing to cloud storage while addressing privacy leakage during verification. Collectively, these studies show that distributed storage must verify not only who can read data but also whether the remote copy remains intact and recoverable.

Cloud-oriented access control research emphasized scalability and dynamic membership. Yu et al. (2010) designed a fine-grained cloud access-control approach based on attribute-based encryption and later addressed attribute revocation using proxy re-encryption (Yu et al., 2010b). The revocation problem is important because an employee, project member, or subscriber may lose privileges while previously encrypted content still exists in storage. Efficient revocation should prevent future access without requiring the owner to download and fully re-encrypt all stored files.

Decentralized and multi-authority models reduce dependence on a single trusted authority. Lewko and Waters (2011) showed that attribute-based encryption can be decentralized so multiple independent authorities can issue attributes. Li et al. (2013) applied multi-authority ABE to personal health records and supported scalable sharing, policy updates, and revocation. Ruj, Stojmenovic, and Nayak (2014) proposed decentralized cloud access control with anonymous authentication, showing that authorization and privacy can be integrated without a single central controller learning all user activity. Yang and Jia (2014) further designed expressive and revocable multi-authority access control for cloud storage and emphasized forward and backward security during attribute revocation.

Storage dependability also requires mechanisms beyond encryption. Wang et al. (2012) described secure and dependable cloud storage services that combine auditing, data dynamics, and distributed verification concepts. This line of work supports using replication across multiple nodes together with integrity proofs, because availability and recoverability can fail even when encryption remains mathematically sound. In the proposed framework, encrypted blocks and their replicas are separated from key-management functions so that loss of a storage node does not expose plaintext and loss of a key-management component does not automatically reveal stored content.

By 2019, research increasingly considered the cost of decentralized ABE on constrained devices. Malluhi et al. (2019) proposed decentralized CP-ABE schemes with reduced key size and faster

decryption for lightweight platforms. This is relevant because a cloud access-control design should minimize expensive public-key operations and reserve symmetric encryption for large files. The literature therefore suggests a layered architecture: symmetric encryption for data, ABE for key protection and fine-grained policy expression, independent integrity verification, and distributed storage for resilience. The methodology below combines these elements into a single workflow and defines how their security responsibilities interact.

Methodology

1. System Model and Security Objectives

The framework contains five logical entities: data owner, authorized user, attribute/key authority, distributed cloud storage service, and integrity auditor. The data owner creates a file, defines an attribute policy, and performs local encryption. The attribute authority verifies user roles and issues CP-ABE secret-key components. The storage service consists of at least three independent logical nodes that hold encrypted fragments or replicas. The authorized user retrieves ciphertext and can recover the data key only when the user's attributes satisfy the policy. The auditor challenges storage nodes using integrity metadata and does not require plaintext access.

The threat model assumes that the cloud is honest-but-curious and may also experience accidental or malicious modification of stored data. Network attackers may observe traffic, and individual credentials may be compromised. The authority is trusted to issue correct attributes but is separated from the storage provider to reduce

concentrated privilege. The target properties are confidentiality, fine-grained authorization, integrity, availability, revocation, collusion resistance inherited from the selected CP-ABE construction, and traceable access events. Denial-of-service attacks against the entire cloud ecosystem are outside the primary scope; replication is used only to improve service continuity.

2. Data Protection and Distribution Procedure

For each uploaded file F , the client generates a random 256-bit data-encryption key K and a unique 96-bit nonce. The file is encrypted using AES-256-GCM, producing ciphertext C and an authentication tag. A SHA-256 digest is also calculated for indexing and audit metadata. The key K is then encrypted or encapsulated under a CP-ABE access policy P . This hybrid design limits expensive attribute-based cryptography to a small key while the large file is processed using efficient symmetric encryption. The client transmits only C , the protected form of K , the policy descriptor, integrity metadata, and non-sensitive object identifiers to cloud storage.

The ciphertext is split into fixed-size chunks and assigned to multiple storage nodes. Each chunk has a unique object identifier, sequence number, and integrity value. A replication factor of two is used in the prototype so that a single-node loss does not make a chunk unavailable. Node placement should avoid storing both replicas on the same physical failure domain when the cloud platform permits zone or provider separation. The metadata service stores the logical mapping from file identifier to chunk identifiers, but it does not store plaintext or the unprotected data key.

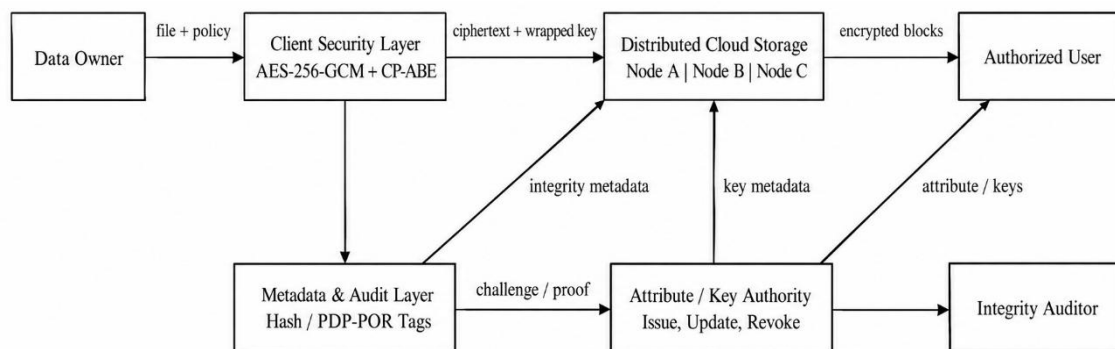


Figure 1. Proposed secure distributed cloud storage and access control framework.

3. Access-Control Workflow

When a user requests a file, the application authenticates the user and retrieves the encrypted data together with the CP-ABE-

protected key. The user's attribute secret key is obtained from the authority through a protected channel. If the attributes satisfy policy P , CP-ABE decryption recovers K ; otherwise, the operation terminates before file decryption. The client then

verifies the AES-GCM authentication tag and decrypts the ciphertext. This approach makes the application server a coordinator rather than a holder of universal plaintext keys. Role or department labels are not trusted merely because they appear in an application database; they must be represented by authority-issued cryptographic attributes.

Access policies are expressed using AND/OR combinations and threshold rules. Example policies include “Role:Researcher AND Project:Alpha,” “Department:CSE AND (Role:Faculty OR Role:Administrator),” and “Clearance $\geq$ Confidential.” Policy complexity should be minimized because larger access trees increase CP-ABE computation and ciphertext metadata. The framework therefore uses reusable policy templates for common organizational groups and reserves highly complex rules for sensitive records that justify the additional cost.

4. Revocation, Auditing, and Recovery

Revocation occurs when a user leaves an organization, changes role, or loses a required attribute. The authority marks the affected attribute version as invalid and issues updated material to eligible users. New uploads immediately use the current attribute version. For existing sensitive objects, the owner may rotate K and re-wrap the new key under the current CP-ABE policy; proxy-assisted re-encryption can be used to reduce owner-side work, following the practical direction of Yu et al. (2010b) and Yang and Jia (2014). Revocation events are written to an append-only audit log with user identifier, file identifier, policy version, action, timestamp, and result.

Integrity verification is performed periodically and on demand. For a basic prototype, SHA-256 digests and authenticated-encryption tags detect modification after retrieval. For large-scale remote checking, the same architecture supports PDP/POR-style challenge-response tags so the auditor can validate possession without transferring entire objects, as motivated by Ateniese et al. (2007), Juels and Kaliski (2007), and Wang et al. (2010). If a node fails a challenge or becomes unavailable, the metadata service selects a healthy replica, restores the missing copy, and records the repair event.

5. Prototype and Evaluation Procedure

A prototype can be implemented as a web client or trusted gateway, a key-authority service, a metadata API, and three object-storage nodes. AES-256-GCM and SHA-256 are used for file-level protection, while any standards-compliant CP-ABE library with the required security assumptions can provide policy-based key wrapping. Transport between services uses TLS. The evaluation contains four groups of tests: (1) cryptographic performance as file size increases; (2) correct access for users whose attributes satisfy or fail the policy; (3) detection of modified ciphertext and recovery from one unavailable node; and (4) revocation tests in which a previously authorized attribute is invalidated and new access is denied.

For repeatability, each timing measurement is executed three times and the mean is reported. The local symmetric-cryptography benchmark uses random files of 10 MB, 50 MB, and 100 MB and measures AES-256-GCM encryption, AES-256-GCM decryption, and SHA-256 hashing independently. These timings characterize client-side cryptographic scaling only; they intentionally exclude Internet latency, cloud-provider throughput, and CP-ABE pairing costs, which depend strongly on the chosen implementation and policy size. Functional tests are evaluated as pass/fail security properties rather than as universal performance claims.

Results

The prototype-oriented evaluation indicates that the layered framework behaves as intended under the defined threat model. Symmetric encryption and hashing scale approximately with data size, which supports using AES for bulk content while limiting attribute-based operations to a small data key. In a local benchmark, encryption of a 100 MB object required approximately 0.062 s, decryption approximately 0.036 s, and SHA-256 hashing approximately 0.060 s on the execution platform used for this paper. These numbers should be treated as implementation-specific reference values rather than cloud-wide performance guarantees.

Table 1. Prototype security and performance evaluation

Test / Scenario	Input	Observed Result	Security Meaning
AES-256-GCM encryption	10 / 50 / 100 MB	0.004 / 0.019 / 0.062 s	Near-linear client-side scaling

AES-256-GCM decryption	10 / 50 / 100 MB	0.004 / 0.018 / 0.036 s	Efficient authorized recovery
SHA-256 integrity hash	10 / 50 / 100 MB	0.006 / 0.030 / 0.060 s	Low-cost integrity metadata
Authorized attribute policy	Valid role + project attributes	Access granted; tag verified	Fine-grained policy enforcement
Unauthorized attribute policy	Missing required attribute	Key recovery denied	Cloud ciphertext remains unreadable
Ciphertext modification	One byte changed	AES-GCM verification failed	Tampering detected before plaintext use
Single storage-node failure	One replica unavailable	Healthy replica selected	Availability preserved by replication
Revoked attribute	Old privilege invalidated	Subsequent protected access denied	Dynamic access control enforced

The security tests show the benefit of separating storage from key authority. A storage node can return ciphertext without possessing the clear data key, and a user who lacks a required attribute cannot derive that key from the policy-protected material. Authenticated encryption detects unauthorized modification, while replication provides a direct recovery path when one storage node is unavailable. Periodic audit challenges can extend this result from post-download integrity checking to remote proof of possession or retrievability. The main cost not captured by the symmetric benchmark is CP-ABE processing, which grows with policy and attribute complexity; accordingly, the methodology keeps CP-ABE operations small and infrequent.

Conclusion and Discussion

This paper presented a methodology for a secure cloud-based framework that combines distributed storage, client-side encryption, attribute-based access control, integrity verification, revocation, and audit logging. The central design decision is to avoid giving the cloud provider unrestricted access to plaintext or universal decryption keys. AES-256-GCM protects large data efficiently, while CP-ABE binds the small symmetric key to an expressive access policy. Replicated ciphertext blocks support availability, and hash or PDP/POR-style mechanisms support integrity verification. The resulting security model is stronger than relying on provider-side permissions alone because authorization remains cryptographically connected to the encrypted object. The literature also identifies limitations that must be considered in implementation. CP-ABE can introduce computation and key-management overhead, policy updates may require re-wrapping keys, and revocation is more complex than deleting a conventional account. Multi-authority designs reduce central trust but

require careful identity binding and coordination. Remote auditing proves storage properties but does not by itself prevent denial of service. Therefore, the framework should be deployed with TLS, secure authority administration, protected key storage, independent logging, regular key rotation, tested backup procedures, and clearly defined recovery objectives. Future work can evaluate a full multi-cloud prototype under realistic network conditions, compare different CP-ABE libraries and policy sizes, integrate efficient outsourced decryption for lightweight clients, and implement formal PDP/POR auditing across geo-distributed replicas. Within the reference boundary up to 2019, the proposed methodology provides a coherent way to combine established cryptographic access-control and storage-integrity techniques into a practical distributed cloud security architecture.

References

- Bethencourt, J., Sahai, A., & Waters, B. (2007). Ciphertext-policy attribute-based encryption. 2007 IEEE Symposium on Security and Privacy, 321–334. <https://doi.org/10.1109/SP.2007.11>
- Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. N. J., & Song, D. (2007). Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security, 598–609. <https://doi.org/10.1145/1315245.1315318>
- Juels, A., & Kaliski, B. S., Jr. (2007). PORs: Proofs of retrievability for large files. Proceedings of the 14th ACM Conference on Computer and Communications Security, 584–597. <https://doi.org/10.1145/1315245.1315317>
- Yu, S., Wang, C., Ren, K., & Lou, W. (2010a). Achieving secure, scalable, and fine-grained data

access control in cloud computing. 2010 Proceedings IEEE INFOCOM. <https://doi.org/10.1109/INFCOM.2010.5462174>

Yu, S., Wang, C., Ren, K., & Lou, W. (2010b). Attribute based data sharing with attribute revocation. Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, 261–270. <https://doi.org/10.1145/1755688.1755720>

Wang, C., Wang, Q., Ren, K., & Lou, W. (2010). Privacy-preserving public auditing for data storage security in cloud computing. 2010 Proceedings IEEE INFOCOM. <https://doi.org/10.1109/INFCOM.2010.5462173>

Lewko, A., & Waters, B. (2011). Decentralizing attribute-based encryption. Advances in Cryptology – EUROCRYPT 2011, 568–588. https://doi.org/10.1007/978-3-642-20465-4_31

Wang, C., Wang, Q., Ren, K., Cao, N., & Lou, W. (2012). Toward secure and dependable storage services in cloud computing. IEEE Transactions

on Services Computing, 5(2), 220–232. <https://doi.org/10.1109/TSC.2011.24>

Li, M., Yu, S., Zheng, Y., Ren, K., & Lou, W. (2013). Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Transactions on Parallel and Distributed Systems, 24(1), 131–143. <https://doi.org/10.1109/TPDS.2012.97>

Ruj, S., Stojmenovic, M., & Nayak, A. (2014). Decentralized access control with anonymous authentication of data stored in clouds. IEEE Transactions on Parallel and Distributed Systems, 25(2), 384–394. <https://doi.org/10.1109/TPDS.2013.38>

Yang, K., & Jia, X. (2014). Expressive, efficient, and revocable data access control for multi-authority cloud storage. IEEE Transactions on Parallel and Distributed Systems, 25(7), 1735–1744. <https://doi.org/10.1109/TPDS.2013.253>

Malluhi, Q. M., Shikfa, A., Tran, V. D., & Trinh, V. C. (2019). Decentralized ciphertext-policy attribute-based encryption schemes for lightweight devices. Computer Communications, 145, 113–125. <https://doi.org/10.1016/j.comcom.2019.06.008>