



An Efficient Data Encryption Framework for Secure Cloud Storage Systems

Kenjiro Rafizadeh

Department of Electrical and Computer Engineering, Siam Delta Engineering Institute, Thailand
kenjiro.rafizadeh@sdei-th.edu

Peer Review Information

Submission: 18 April 2016

Revision: 09 May 2016

Acceptance: 26 May 2016

Keywords

Cloud Storage, Data Encryption, AES-GCM, Key Management, Access Control

Abstract

Cloud storage reduces local infrastructure cost and improves data availability, but outsourcing sensitive information to infrastructure outside the data owner's direct control creates persistent confidentiality, integrity, and access-control risks. This paper presents a methodology for an efficient hybrid data-encryption framework designed to secure files before they are transferred to cloud storage. The framework separates bulk-data encryption from key protection: file contents are protected with AES-256 in Galois/Counter Mode (AES-GCM), while the per-file symmetric key is wrapped with RSA-2048 using OAEP. Authenticated encryption detects unauthorized ciphertext modification, and an independent policy/key service prevents the cloud-storage provider from receiving plaintext encryption keys. Fine-grained authorization concepts are incorporated at the key-release layer so that storage access and decryption permission are treated as separate decisions. The evaluation methodology combines security-property analysis with a reproducible prototype benchmark across 1, 5, 10, and 25 MB files. Results show nearly linear processing time with file size and negligible storage expansion because public-key cryptography is applied only to the small data-encryption key. The framework therefore provides a practical balance between confidentiality, integrity, controlled key distribution, and computational efficiency for cloud storage applications.

Introduction

Cloud computing changed storage from a locally controlled asset into an on-demand service in which data, applications, and infrastructure may be operated by a third party. The economic and operational advantages are substantial, yet cloud adoption has consistently been accompanied by concerns over data confidentiality, privacy, tenant isolation, and loss of direct physical control. Subashini and Kavitha identified data storage and transmission security among the major risks across cloud service-delivery models [1], while Zissis and Lekkas emphasized that cloud security must preserve trust even when traditional organizational boundaries are weakened [2]. Chen and Zhao similarly treated

confidentiality and privacy throughout the data life cycle as primary barriers to wider cloud adoption [3].

Encryption is a direct response to the confidentiality problem, but simply applying a cipher to a file is insufficient for a usable cloud-storage system. A complete design must decide where encryption takes place, how keys are generated and protected, how legitimate users obtain keys, how unauthorized users are revoked, and how integrity is verified after data has been outsourced. If the same cloud provider stores both ciphertext and the keys needed to decrypt it, compromise of the provider can defeat the protection. Conversely, encrypting every

large file with public-key cryptography is computationally inefficient. The central engineering problem is therefore to combine fast symmetric encryption with strong key protection and policy-aware key release.

This study proposes a client-side hybrid framework in which each file is encrypted with a unique symmetric key before upload. The cloud stores only authenticated ciphertext and non-secret metadata; the symmetric key is separately protected with a public-key mechanism and released only after an authorization decision. The methodology focuses on four objectives: (i) confidentiality of outsourced data, (ii) detection of unauthorized modification, (iii) separation of storage access from decryption authority, and (iv) low processing and storage overhead. The contribution is not a new cryptographic primitive; rather, it is a systems-level integration of established cryptographic mechanisms into an efficient workflow suitable for secure cloud storage.

Literature Review

Early work established the principle that cloud storage should remain secure even when the storage provider is not fully trusted. Kamara and Lauter described cryptographic cloud-storage architectures that place cryptography between the user and public-cloud infrastructure [4]. Complementary research addressed integrity: Wang et al. developed a secure and dependable cloud-storage model with mechanisms for checking outsourced data [5], and later demonstrated privacy-preserving public auditing that permits an external auditor to verify storage integrity without learning user data [6]. These studies show that confidentiality and integrity should be designed together rather than treated as isolated controls.

Fine-grained cryptographic authorization also became central to outsourced-data protection. Goyal et al. introduced key-policy attribute-based encryption (KP-ABE) for controlling encrypted data with attribute-based rules [7]. Bethencourt et al. shifted the access policy into the ciphertext through ciphertext-policy ABE (CP-ABE), allowing the encryptor to define which attribute combinations are permitted to decrypt [8]. Yu et al. applied attribute-based concepts to cloud computing to achieve scalable and fine-grained data access control [9]. Later, Liang and Susilo combined searchable attributes and secure data sharing for outsourced cloud data [10]. These mechanisms offer powerful policy enforcement, although pairing-based operations can be expensive for routine bulk-file encryption.

System-oriented frameworks attempted to separate security management from storage management. Vu et al. proposed Secure Cloud Storage as Data Protection as a Service, supporting automated encryption and fine-grained policies across multi-cloud environments [11]. Bellare et al. examined message-locked encryption for secure deduplication, highlighting the tension between storage efficiency and confidentiality when identical files are intentionally mapped to related encryption material [12]. Liu et al. further demonstrated that access structures themselves may reveal sensitive information and introduced partially hidden CP-ABE policies [13]. Collectively, the literature suggests that a practical storage framework should use efficient symmetric encryption for large objects while isolating keys and authorization logic from the untrusted storage plane.

Table 1. Representative pre-2017 studies relevant to secure cloud storage

Study	Core mechanism	Contribution to this methodology	Main limitation
Kamara & Lauter [4]	Cryptographic storage	Client-side protection against untrusted cloud storage	High-level architecture
Wang et al. [6]	Public auditing	Integrity verification without exposing plaintext	Adds auditing complexity
Yu et al. [9]	ABE access control	Fine-grained cryptographic authorization	Pairing overhead
Vu et al. [11]	DPaaS framework	Separates protection logic from storage management	Framework deployment complexity
Liu et al. [13]	Hidden-policy CP-ABE	Protects sensitive access-policy values	Higher cryptographic cost

Methodology

1. Framework Design and Threat Model

The proposed system contains five logical actors: data owner, client security gateway, policy/key service, cloud-storage provider, and authorized

data user. The cloud-storage provider is assumed to be honest-but-curious: it correctly stores and returns objects but may attempt to inspect contents. Network traffic is assumed to be protected by standard secure transport, while the methodology focuses on protection of data at rest and during key release. The policy/key service is logically separated from the object store so that compromise of storage credentials alone does not reveal the plaintext encryption keys. This separation follows the broader cryptographic-storage principle of minimizing trust in the storage provider [4] and the policy automation direction described by Vu et al. [11].

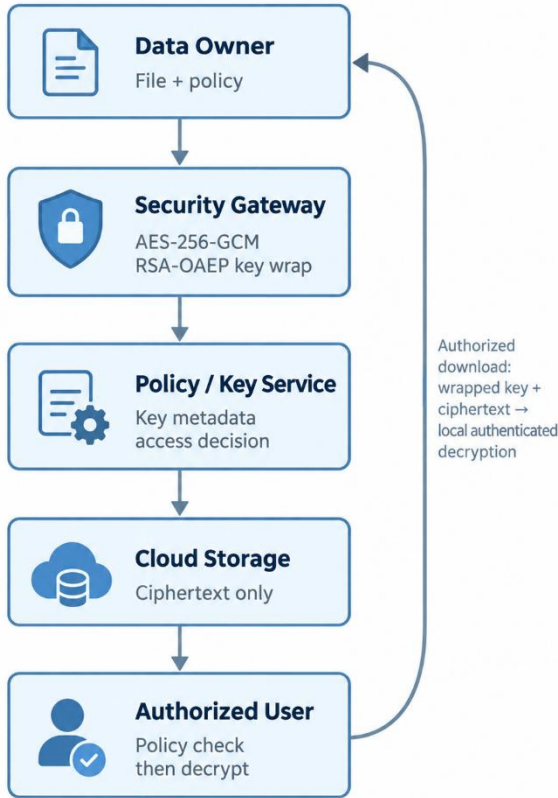


Figure 1. Proposed hybrid encryption framework for secure cloud storage.

2. Encryption and Upload Procedure

For every uploaded object, the gateway generates an independent 256-bit random data-encryption key K and a 96-bit nonce N . The plaintext file D is encrypted locally using AES-256-GCM, producing ciphertext C and an authentication tag. The operation can be represented as $C = \text{AES-GCM}(K, N, D, \text{AAD})$, where AAD contains immutable non-secret metadata such as file identifier and version. AES-GCM is selected because one primitive provides both confidentiality and integrity, avoiding a separate encrypt-then-MAC pipeline. The small key K , rather than the full file, is then encrypted with the authorized key service's RSA-2048 public key

using OAEP with SHA-256: $W = \text{RSA-OAEP}(PK, K)$. Only $\{C, N, W, \text{AAD}\}$ are stored or associated with the cloud object.

This hybrid construction is efficient because expensive public-key operations are constant with respect to file size. Attribute-based encryption could replace or supplement RSA key wrapping when policy complexity requires cryptographic fine-grained access control, as demonstrated by Goyal et al. [7], Bethencourt et al. [8], and Yu et al. [9]. In the present framework, however, bulk data remain under symmetric encryption and policy evaluation controls release or re-wrapping of the data key. This preserves the systems efficiency of hybrid encryption while retaining a migration path to richer policy mechanisms, including hidden-policy methods such as those in [13].

3. Authorization, Download, and Revocation

A user requesting a file first authenticates to the policy/key service. The service evaluates user identity, role or attributes, file policy, and current authorization state. If access is denied, the cloud ciphertext remains computationally unusable even if the user possesses a storage URL. If access is granted, the service unwraps K inside the trusted key-management boundary or re-wraps it for the requesting user. The authorized client then downloads C and performs local authenticated decryption. Any modification to ciphertext, nonce, or authenticated metadata causes tag verification to fail. Revocation is implemented by denying future key releases; high-risk objects can additionally be re-keyed by generating K' and re-encrypting the object. The design thus distinguishes storage authorization from cryptographic authorization, a distinction consistent with secure fine-grained data sharing research [9,10].

4. Evaluation Metrics

The framework is evaluated using both security and efficiency metrics. Security analysis checks whether the design provides confidentiality against the storage provider, integrity against ciphertext modification, per-file key isolation, policy-controlled key release, and resistance to key exposure through storage compromise. Efficiency is measured by encryption time (T_{enc}), decryption time (T_{dec}), processing throughput, and storage expansion. For a plaintext of size S and stored encrypted object size S_e , storage overhead is calculated as $\text{Overhead}(\%) = ((S_e - S)/S) \times 100$. Processing throughput is S/T . Public auditing and proof-of-possession techniques from [5,6] may be added

when independent remote-integrity assurance is required beyond the local AES-GCM authenticity check.

5. Experimental Setup

A proof-of-concept benchmark was executed using Python 3 with the OpenSSL-backed cryptography library on an x86-64 AMD EPYC 9V74 environment. Test objects of 1, 5, 10, and 25 MB were generated from random bytes. For every run, a fresh AES-256-GCM key and 96-bit nonce were created, the file was encrypted, and the AES key was wrapped with a newly generated RSA-2048 public key using OAEP/SHA-256. Decryption unwrapped the key and verified the AES-GCM tag. Each file size was processed seven times, and the median was retained to reduce short-duration timing noise. The results are intended to show scaling behavior of the framework; absolute values will vary across

processors, cryptographic libraries, storage devices, and cloud network paths.

Results

The prototype results in Table 2 show that processing time increased approximately with object size, while storage expansion rapidly approached zero as the file became larger. This behavior is expected because AES-GCM processes the entire payload, whereas the RSA operation handles only a 32-byte symmetric key. For the 25 MB object, median encryption required about 9.41 ms and decryption about 8.92 ms in the benchmark environment, corresponding to multi-gigabyte-per-second in-memory cryptographic throughput. The constant encrypted-key, nonce, and authentication-tag metadata produced only about 0.0011% storage expansion for the 25 MB test file.

Table 2. Prototype performance of the proposed hybrid encryption workflow

File (MB)	Encrypt (ms)	Decrypt (ms)	Enc. throughput (MB/s)	Dec. throughput (MB/s)	Storage overhead (%)
1	0.37	0.36	2677	2783	0.0271
5	1.89	1.82	2647	2744	0.0054
10	3.53	1.72	2832	5818	0.0027
25	9.41	8.92	2657	2801	0.0011

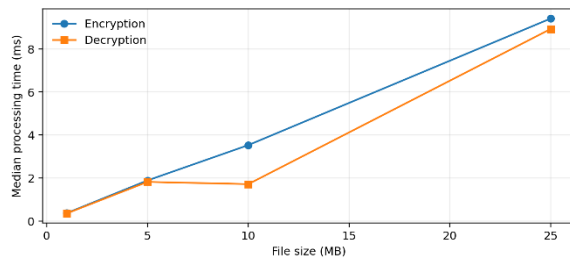


Figure 2. Median encryption and decryption time as file size increases.

Security-property analysis also indicates that the framework reduces the consequences of a single storage-layer compromise. A storage attacker obtains ciphertext but not an unwrapped data key. Modification of stored content is detected by authenticated decryption. Unique per-file keys limit key reuse across objects, and policy-controlled release supports immediate denial of future decryptions after a user is revoked. These properties address the core confidentiality and integrity concerns highlighted in cloud-security surveys [1-3]. They also complement remote-auditing approaches [5,6], which can be integrated when organizations need evidence that the provider retains complete outsourced files.

The principal trade-off is key-management complexity. A dedicated key service becomes security-critical and must itself be hardened, backed up, and audited. Re-encrypting very large objects after revocation can also be expensive; in many deployments, re-wrapping small data keys is preferable when the threat model permits it. ABE-based policies offer stronger cryptographic binding between attributes and decryption authority [7-10], but they generally impose higher computation and metadata costs than the RSA key-wrap layer used in this benchmark. The selected architecture therefore favors common enterprise cloud-storage workloads where fast bulk encryption and centralized policy enforcement are the priority.

Conclusion and Discussion

This methodology demonstrates that secure cloud storage can be constructed efficiently by separating bulk data protection, key protection, and authorization. AES-256-GCM protects large objects with authenticated symmetric encryption, while RSA-OAEP protects only the small per-file key. The resulting workflow prevents the cloud object store from serving as both data custodian and key custodian, thereby reducing trust concentration. The prototype

benchmark shows near-linear processing cost with file size and negligible storage expansion, supporting the practical use of client-side encryption for typical cloud objects. The design is consistent with earlier cryptographic cloud-storage work [4], secure-storage and auditing research [5,6], and policy-oriented protection mechanisms [7-11]. Future implementations can extend the framework with hardware-backed key protection, independent public auditing, CP-ABE for decentralized fine-grained policy enforcement, or carefully designed encrypted deduplication informed by message-locked encryption [12]. Policy privacy should also be considered where authorization rules themselves reveal sensitive information, as shown by Liu et al. [13]. Overall, the framework offers a balanced methodology for confidentiality, integrity, access control, and computational efficiency without requiring the storage provider to be fully trusted.

References

- S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1-11, 2011. DOI: 10.1016/j.jnca.2010.07.006.
- D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583-592, 2012. DOI: 10.1016/j.future.2010.12.006.
- D. Chen and H. Zhao, "Data Security and Privacy Protection Issues in Cloud Computing," 2012 International Conference on Computer Science and Electronics Engineering, vol. 1, pp. 647-651, 2012. DOI: 10.1109/ICCSEE.2012.193.
- S. Kamara and K. Lauter, "Cryptographic Cloud Storage," in *Financial Cryptography and Data Security*, LNCS, pp. 136-149, 2010. DOI: 10.1007/978-3-642-14992-4_13.
- C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 220-232, 2012. DOI: 10.1109/TSC.2011.24.
- C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362-375, 2013. DOI: 10.1109/TC.2011.245.
- V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 89-98, 2006. DOI: 10.1145/1180405.1180418.
- J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," 2007 IEEE Symposium on Security and Privacy, pp. 321-334, 2007. DOI: 10.1109/SP.2007.11.
- S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *IEEE INFOCOM 2010*, pp. 1-9, 2010. DOI: 10.1109/INFCOM.2010.5462174.
- K. Liang and W. Susilo, "Searchable Attribute-Based Mechanism With Efficient Data Sharing for Secure Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1981-1992, 2015. DOI: 10.1109/TIFS.2015.2442215.
- Q. H. Vu, M. Colombo, R. Asal, A. Sajjad, F. A. El-Moussa, and T. Dimitrakos, "Secure Cloud Storage: A Framework for Data Protection as a Service in the Multi-Cloud Environment," 2015 IEEE Conference on Communications and Network Security, pp. 638-642, 2015. DOI: 10.1109/CNS.2015.7346879.
- M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-Locked Encryption and Secure Deduplication," in *Advances in Cryptology - EUROCRYPT 2013*, pp. 296-312, 2013. DOI: 10.1007/978-3-642-38348-9_18.
- L. Liu, J. Lai, R. H. Deng, and Y. Li, "Ciphertext-Policy Attribute-Based Encryption with Partially Hidden Access Structure and Its Application to Privacy-Preserving Electronic Medical Record System in Cloud Environment," *Security and Communication Networks*, vol. 9, pp. 4897-4913, 2016. DOI: 10.1002/sec.1663.