Proceedings of CONF-MLA 2026 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2026.LD37597

Encryption and Key Management of Personal Data in Cloud Computing: The Case of Amazon S3 and AWS KMS

Xuancheng Bao

Hangzhou Number 4 High School, Hangzhou, China
xcheng08@163.com

Abstract. Cloud computing has enabled individuals and organizations to store growing volumes of personal data efficiently, but it has also reduced users' direct control over storage infrastructure and increased exposure to unauthorized access. This paper examines the protection of cloud-based personal data through envelope encryption and key management, using Amazon S3 and AWS Key Management Service (AWS KMS) as a case study. It addresses the following research question: To what extent can the S3–KMS encryption mechanism reduce personal-data leakage risks, and what factors limit its practical effectiveness? Based on a review of academic literature and AWS technical documentation, the paper analyzes the roles of data encryption keys (DEKs), key encryption keys (KEKs), access permissions, key rotation, and audit logs. The analysis finds that SSE-KMS can reduce the risk that copied stored data will directly disclose personal information by separating encrypted objects from the key encryption keys (KEKs) needed to recover their data encryption keys (DEKs). However, encryption alone cannot prevent breaches caused by stolen credentials, excessive permissions, improper key deletion, or inadequate log monitoring. The paper concludes that effective cloud privacy protection requires a coordinated framework combining encryption, key lifecycle governance, least-privilege access control, and continuous auditing.

Keywords: Cloud Computing, Personal Data Security, Envelope Encryption, AWS, KMS

1. Introduction

As cloud computing has developed, growing volumes of personal data have been stored and processed on cloud platforms. Compared with local storage, cloud storage offers elastic capacity and convenient access, but it also changes how data owners exercise control over their information. After data are uploaded, their physical location, underlying infrastructure, and some management processes are controlled by the cloud service provider. Unauthorized access to identity records, user files, or transaction records may consequently cause privacy violations and financial loss. Existing research identifies data security and privacy as major concerns affecting the adoption of cloud services and emphasizes the need to combine encryption with access control and identity authentication [1].

Against this background, this paper examines the encryption and key-management mechanisms used to protect personal data in cloud storage, using Amazon Simple Storage Service (Amazon S3)

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.LD37597

and AWS Key Management Service (AWS KMS) as a case study. It addresses the following research question: How does the envelope encryption mechanism based on Amazon S3 and AWS KMS protect personal data, and what factors may influence its practical effectiveness? Using a literature-review approach, the paper first discusses personal-data leakage risks in cloud environments and explain the relationship between envelope encryption and key management. It then examines the encryption process, protective functions, and limitations of S3 and KMS before presenting coordinated protection recommendations. This study helps clarify the connection between cryptographic technology and security management in cloud-based personal-data protection.

2. Personal data leakage in the cloud

Personal data include information that can identify an individual independently or in combination with other information. They include structured identity records and unstructured materials such as photographs, documents, backups, and application logs. Providers may replicate these materials across devices or data centers for backup, disaster recovery, and availability. Although replication improves reliability, it increases the systems, identities, and administrative processes involved. Protection must therefore address not only data loss but also the unauthorized reading, copying, modification, and distribution of data.

Cloud data breaches may result from both external attacks and internal management failures. Attackers may steal passwords, exploit application vulnerabilities, or obtain access credentials. Internal risks may arise from excessive permissions, improper actions by administrators, or misconfigured policies. For example, an application that only needs access to a small group of files may mistakenly be granted permission to read an entire storage bucket. If its credentials are compromised, the attacker may acquire far more information than the application requires. Singh, Jeong, and Park observe that cloud security threats extend across identity management, data storage, networks, and applications [2]. A single safeguard is therefore unlikely to address the full range of risks.

Cloud storage also separates data ownership from infrastructure control. Customers may retain ownership of their information, while the provider operates the infrastructure. Xiao notes that cloud storage reduces the owner's direct physical control over the data, access control, and integrity verification [1]. Sun and colleagues likewise explain that confidentiality, integrity, and availability must be considered together [3].

Encryption converts plaintext into ciphertext that is difficult for an unauthorized party to understand. If storage media, backup files, or transmitted content are obtained illegally, encryption can prevent immediate disclosure. However, it mainly protects the content of data and does not independently decide whether an access request serves a legitimate business purpose. An attacker who steals an authorized account or application credential may still obtain plaintext through the normal service interface. For this reason, evaluating cloud protection requires an examination of how keys are generated, stored, authorized, used, rotated, and audited.

3. Envelope encryption and key management

Symmetric encryption is commonly used to protect cloud data because it is efficient for large files, images, and backups. The key that directly encrypts the information is generally called a data encryption key (DEK). An application uses the DEK to transform personal data into ciphertext before or during storage. As long as the DEK remains unavailable to unauthorized parties, a copied ciphertext object should not disclose its original content.Key use nevertheless creates a management

DOI: 10.54254/2755-2721/2026.LD37597

problem. If one DEK protects extensive data, its compromise may expose many records. If every object receives a separate key, the organization must manage a rapidly growing key collection. Sun and colleagues identify related challenges involving key management, fine-grained authorization, querying, and modification [3]. Cloud encryption is therefore both a cryptographic and governance problem.

Envelope encryption addresses this tension through a hierarchy of keys. A DEK encrypts the data, while a higher-level key encryption key (KEK) encrypts the DEK. The ciphertext and encrypted DEK may be stored together, but the KEK remains in a separate, tightly controlled key-management system. During decryption, the system first verifies the caller's identity and permissions, uses the KEK to recover the DEK, and then uses the DEK to recover the data. This design retains the efficiency of symmetric encryption while centralizing control over higher-level keys. Even if an attacker obtains both the ciphertext and the encrypted DEK, the information remains protected unless the attacker gains access to the KEK. NIST explains that cloud key management is especially complex because ownership and infrastructure control are divided between the cloud customer and provider [4].

The effectiveness of envelope encryption depends on the entire key lifecycle. Keys should be generated with sufficient randomness, protected against prolonged exposure in plaintext, and restricted to authorized users, programs, and operations. Organizations should also establish appropriate rotation, disabling, deletion, and audit procedures. Research on cloud privacy accordingly emphasizes that encryption should work with access control, revocation, tracing, and trust-management mechanisms rather than being deployed in isolation [5]. A study of searchable encryption by Yang and colleagues further illustrates that theoretical cryptographic security does not automatically guarantee secure deployment; inappropriate implementation or use can introduce additional risks [6].

4. Encryption, protection, and limitations of Amazon S3 and AWS KMS

Amazon S3 is an object-storage service in which documents, images, application data, and backups are stored as objects in buckets. AWS KMS creates and controls cryptographic keys and uses key policies to regulate access by users, applications, and AWS services. Campagna and Gueron explain that KMS was designed to provide security, availability, and performance in a cloud environment containing many users, keys, and concurrent requests [7]. Amazon S3 automatically applies server-side encryption to new objects, using S3-managed keys (SSE-S3) by default. Customers who require more detailed control over key permissions and management may instead select server-side encryption with AWS KMS keys (SSE-KMS) [8].

Under SSE-KMS, S3 does not use the KMS key to encrypt an entire object directly. When an object is uploaded, S3 requests a plaintext DEK and an encrypted copy of that DEK from KMS. KMS generates the DEK, encrypts it under the selected KMS key, and returns both versions to S3. S3 encrypts the object with the plaintext DEK and removes that plaintext key from memory as soon as practicable. It stores the encrypted object together with the encrypted DEK, while KMS controls the higher-level key encryption key (KEK) [9]. When an authorized user downloads the object, S3 sends the encrypted DEK to KMS. After evaluating the request and permissions, KMS decrypts the DEK and returns its plaintext form to S3, which decrypts the object and again removes the plaintext DEK after use.

This mechanism offers several protections. A party that copies the ciphertext and encrypted DEK still requires authorization to use the relevant KMS key. Permissions for storing an object can also be separated from permissions for decrypting it, allowing an organization to distinguish uploaders,

DOI: 10.54254/2755-2721/2026.LD37597

readers, storage administrators, and key administrators [9]. Eligible customer-managed symmetric KMS keys support automatic or on-demand rotation, while KMS retains the material needed to decrypt earlier data [10]. Key-management and cryptographic API operations can also be recorded through AWS CloudTrail, supporting investigation of which identity performed an operation and when [11].

These controls do not eliminate every route to disclosure. SSE-KMS primarily protects data at rest. If an attacker steals an account or application credential that already has permission to access the S3 object and use the KMS key, the resulting request may be processed as authorized. Poorly designed key policies can similarly undermine the encryption hierarchy. Giving many users or applications broad decryption permissions enlarges the attack surface, while failure to separate administrative duties may allow one compromised identity to control both data access and key use. KMS provides tools for authorization, but it cannot determine the customer's legitimate business requirements or correct an unnecessarily permissive policy.

Key rotation also has important limits. Rotating KMS key material does not automatically rotate existing DEKs or re-encrypt previously stored S3 objects [10]. It therefore does not remove the risk created by a DEK that has already been exposed. Conversely, mistakenly disabling or deleting a required KMS key may make associated objects unreadable. Finally, logs provide value only when they are retained, reviewed, and connected to alerts and incident-response procedures. CloudTrail can record KMS events, but merely enabling logs does not prevent a breach. Effective monitoring should detect unusual volumes of decryption, unexpected cross-account activity, unfamiliar identities, and changes to key policies.

5. Coordinated protection

Cloud protection should begin with data classification. Routine information, sensitive personal data, and high-impact identity records should not necessarily share identical access and key arrangements. Sensitive categories may justify customer-managed KMS keys separated by data type, application, or unit. Organizations should also block unintended public access to S3 buckets and use bucket policies to require approved encryption settings.

The principle of least privilege should apply independently to S3 permissions and KMS permissions. Permission to view an object should not automatically provide access to every encryption key, and authority to administer a key should not automatically provide access to all personal data. Key administrators, data administrators, and application identities should receive only the permissions required for their duties. Unused accounts, roles, credentials, and grants should be reviewed and removed regularly.

Organizations must also manage keys throughout their lifecycles. Each key should have a documented purpose, owner, and data scope. Rotation plans should distinguish rotation of KMS key material from replacement of DEKs and re-encryption of stored objects. Before a key is disabled or scheduled for deletion, affected objects and backups should be identified to avoid accidental data loss. If compromise is suspected, response may require restricting permissions, replacing credentials, and re-encrypting affected data rather than relying on routine rotation alone.

6. Conclusion

In conclusion, Amazon S3 and AWS KMS implement a layered relationship between data encryption keys (DEKs) and key encryption keys (KEKs). This envelope-encryption structure can reduce the likelihood that copied stored data will directly reveal personal information while also

DOI: 10.54254/2755-2721/2026.LD37597

supporting centralized permissions, rotation, and audit records. Its protections, however, depend on correct configuration and continuous governance. Strong algorithms cannot prevent misuse by an authorized but compromised identity, excessive permissions, mistaken key deletion, or neglected logs. Cloud privacy protection should therefore combine data encryption, key governance, access control, and active auditing. Because this paper relies on published literature and AWS documentation rather than platform experiments, future research could compare KMS implementations across cloud providers or experimentally evaluate how different permission configurations affect exposure risk.

References

[1] Xiao, R. (2014). Research progress on data privacy protection in cloud computing. *Journal on Communications, 35*(12), 168–177. https://doi.org/10.3969/j.issn.1000-436X.2014.12.020

[2] Singh, S., Jeong, Y. S., & Park, J. H. (2016). A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications, 75*, 200–222. https://doi.org/10.1016/j.jnca.2016.09.002

[3] Sun, Y., Zhang, J., Xiong, Y., & Zhu, G. (2014). Data security and privacy in cloud computing. *International Journal of Distributed Sensor Networks, 10*(7), Article 190903. https://doi.org/10.1155/2014/190903

[4] Chandramouli, R., Iorga, M., & Chokhani, S. (2013). *Cryptographic key management issues and challenges in cloud services* (NISTIR 7956). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.7956

[5] Sun, P. J. (2020). Security and privacy protection in cloud computing: Discussions and challenges. *Journal of Network and Computer Applications, 160*, Article 102642. https://doi.org/10.1016/j.jnca.2020.102642

[6] Yang, N., Jin, Y., Liu, D., & Peng, N. (2015). Security mechanisms and application pitfalls of searchable encryption technology in cloud environments. *Application Research of Computers, 32*(8), 2254–2260. https://doi.org/10.3969/j.issn.1001-3695.2015.08.003

[7] Campagna, M., & Gueron, S. (2019). Key management systems at the cloud scale. *Cryptography, 3*(3), Article 23. https://doi.org/10.3390/cryptography3030023

[8] Amazon Web Services. (n.d.). Setting default server-side encryption behavior for Amazon S3 buckets. Retrieved August 21, 2026, from https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-encryption.html

[9] Amazon Web Services. (n.d.). Using server-side encryption with AWS KMS keys (SSE-KMS). Retrieved August 21, 2026, from https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html

[10] Amazon Web Services. (n.d.). Rotate AWS KMS keys. Retrieved August 21, 2026, from https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html

[11] Amazon Web Services. (n.d.). Logging AWS KMS API calls with AWS CloudTrail. Retrieved August 21, 2026, from https://docs.aws.amazon.com/kms/latest/developerguide/logging-using-cloudtrail.html