



A Secure Data Transmission Framework for Cloud-Based Communication Systems

Jovencio Qureshi-Haq

Department of Electrical and Computer Engineering, Visayan Maritime Polytechnic University, Philippines
jovencio.qureshi.haq@vmpu-ph.net

Peer Review Information	Abstract
<i>Submission: 25 Oct 2015</i> <i>Revision: 19 Nov 2015</i> <i>Acceptance: 30 Dec 2015</i> Keywords <i>Cloud Security, Secure Data Transmission, Authenticated Encryption, Hybrid Cryptograph, Access Control</i>	Cloud-based communication allows organizations to exchange and process data through elastic, shared infrastructure, but it also extends the security boundary beyond the direct control of the data owner. This paper presents a methodology for a secure data transmission framework that protects confidentiality, integrity, authenticity, and controlled access across client-to-cloud and cloud-to-client communication. The framework combines authenticated symmetric encryption for bulk data, asymmetric encryption for session-key protection, identity and policy verification, replay-resistant metadata, and audit logging. The design is motivated by established cloud-security concerns including multi-tenancy, outsourced storage, insecure interfaces, cross-domain trust, and loss of direct data possession [1]-[5]. A controlled prototype was used to measure the cryptographic overhead of AES-256-GCM with RSA-2048/OAEP key encapsulation for 1 MB, 5 MB, and 10 MB payloads. The measured results show millisecond-level encryption/decryption cost on a modern x86-64 platform, while ciphertext expansion remains negligible for ordinary file transfers. Tamper tests also show that modified ciphertext is rejected by the authenticated-encryption layer. The resulting methodology provides a practical, layered approach suitable for cloud messaging, file exchange, and service-to-service communication where security must remain independent of the trustworthiness of the transport or storage provider.

Introduction

Cloud computing changed communication architectures by allowing applications, storage, and processing to operate on remotely provisioned infrastructure. The economic and scalability advantages are substantial, but the model also moves information outside the traditional organizational perimeter. Subashini and Kavitha identified data transmission, storage, application, and third-party security as recurring concerns across cloud service-delivery models [1]. Jensen et al. further emphasized that cloud services create cross-domain technical security problems because data and control functions are exposed through Internet-facing interfaces [2].

The threat is not limited to communication links. Virtualized multi-tenant infrastructure can create information-leakage paths between co-resident workloads, as demonstrated by Ristenpart et al. [3]. Therefore, transport encryption alone is insufficient: a secure design should protect the payload even when cloud infrastructure or intermediate storage is treated as only partially trusted. Cryptographic cloud-storage architectures similarly recommend client-side cryptographic protection so that outsourced providers need not be trusted with plaintext data [4].

This paper develops a step-by-step framework in which each transmitted object is encrypted

before it enters the cloud domain, its short-lived symmetric key is separately protected, access is verified, and every accepted or rejected security event is auditable. The objective is not to introduce a new primitive, but to integrate mature mechanisms into a reproducible transmission methodology that can be evaluated for security coverage and computational overhead.

Literature Review

Cloud security literature before and including 2015 consistently shows that confidentiality, integrity, trust, and accountability must be treated together. Zissis and Lekkas proposed a security model using public-key infrastructure, single sign-on, and directory services to preserve authentication, integrity, and confidentiality across cloud entities [5]. Chen and Zhao analyzed data security and privacy through the cloud-data life cycle and noted that openness, multi-tenancy, and dynamic resource allocation weaken assumptions used in conventional enterprise boundaries [6]. These studies support a framework in which cryptographic protection accompanies the data rather than being bound only to a specific network segment.

Integrity verification is especially important after data has left the owner's system. Ateniese et al. introduced provable data possession, enabling a client to obtain evidence that an untrusted store still possesses outsourced information without downloading the complete file [7]. Juels and Kaliski developed proofs of retrievability for

similar outsourced-storage assurance [8]. Although the present framework focuses on transmission rather than long-term archival auditing, these contributions motivate explicit integrity evidence and audit events at the cloud boundary.

Wang et al. later demonstrated privacy-preserving public auditing for cloud storage, showing that integrity verification can be delegated without revealing the underlying data to the auditor [9]. Fine-grained authorization can also be cryptographically bound to data. Bethencourt, Sahai, and Waters introduced ciphertext-policy attribute-based encryption (CP-ABE), in which encrypted content is decryptable only by users whose attributes satisfy the selected access policy [10]. For this methodology, a simpler identity/policy gate is used in the prototype, while CP-ABE is retained as a deployment extension for environments requiring cryptographically enforced, attribute-level access control.

By 2015, Ali, Khan, and Vasilakos summarized cloud-security challenges as a combination of conventional network/application risks and cloud-specific issues introduced by virtualization, multi-tenancy, shared resources, and third-party control [11]. Their observation that isolated point solutions create operational complexity motivates the integrated structure proposed here. Table 1 summarizes how the literature informs the framework.

Table 1. Literature-to-Framework Mapping

Reference	Main Contribution	Use in Proposed Methodology
Subashini & Kavitha (2011) [1]	Security risks across SaaS/PaaS/IaaS	Treat transmission, storage and application security as linked.
Jensen et al. (2009) [2]	Cross-domain and interface security	Authenticate endpoints and protect cloud-facing interfaces.
Ristenpart et al. (2009) [3]	Co-residency / information leakage	Do not expose plaintext to shared infrastructure.
Kamara & Lauter (2010) [4]	Client-side cryptographic cloud storage	Encrypt before outsourcing; separate provider trust from data secrecy.
Zissis & Lekkas (2012) [5]	PKI and trust services	Use key management and identity assurance.
Ateniese; Juels & Kaliski [7],[8]	Remote integrity evidence	Record verifiable integrity/audit outcomes.
Wang et al. (2013) [9]	Privacy-preserving auditing	Support auditability without revealing plaintext.
Bethencourt et al. (2007) [10]	Fine-grained encrypted access control	Allow policy-based authorization extensions.
Ali et al. (2015) [11]	Integrated cloud-security survey	Layer complementary controls instead of relying on one mechanism.

Methodology

The methodology follows a layered security model. Its central principle is that plaintext and reusable secret keys should not be exposed to the cloud service. A fresh data-encryption key is generated for each transmission, the payload is encrypted with authenticated encryption, the session key is protected with the receiver's

public key, and metadata is checked before decryption. The cloud transports and may temporarily store only ciphertext, protected key material, and non-sensitive routing metadata. This design follows the client-side protection philosophy described by Kamara and Lauter [4] and the PKI-oriented trust approach discussed by Zissis and Lekkas [5].

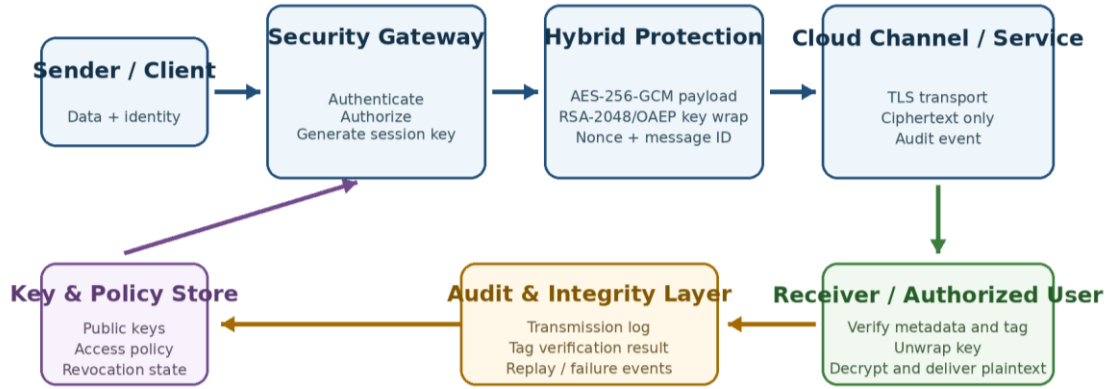


Figure 1. Proposed secure data transmission framework.

1. Security Objectives and Threat Model

The framework targets four primary objectives: confidentiality of payload content, integrity and authenticity of received ciphertext, authorization of the receiver, and traceability of security events. The cloud network, intermediate storage, and neighboring tenants are considered potentially observable or compromised. End-user private keys and the local security gateway are assumed to be protected. Denial-of-service is logged but is outside the main cryptographic scope. This threat model reflects cloud risks documented for Internet-facing services and multi-tenant infrastructure [2], [3], [11].

2. Cryptographic Components

AES-256-GCM is used for bulk data because a symmetric cipher is suitable for high-volume payloads and GCM produces an authentication tag that detects modification. RSA-2048 with OAEP and SHA-256 is used only to encrypt the 256-bit AES session key, preventing expensive public-key operations over the complete payload. A 96-bit random nonce is generated for every GCM operation. A unique message identifier, timestamp, sender identity, receiver identity, algorithm identifier, nonce, and wrapped key are bound to the transmission record. The receiver accepts plaintext only after authentication-tag verification succeeds.

Step-by-Step Transmission Procedure

Step 1 - Identity and policy verification. The sender authenticates to the security gateway. The gateway checks whether the sender may transmit to the stated receiver and whether the receiver is active. In deployments requiring cryptographically enforced fine-grained policies, the authorization layer may be replaced or complemented by CP-ABE concepts [10].

Step 2 - Session-key generation. A cryptographically secure random 256-bit AES key K_s and a fresh 96-bit nonce N are generated for the transmission. Keys are short-lived and are not reused across messages.

Step 3 - Payload encryption and authentication. For plaintext P and associated metadata, A , the sender computes $C, Tag = AES-GCM(K_s, N, P, A)$. The associated data can contain non-secret identifiers that must be authenticated but do not need encryption.

Step 4 - Session-key protection. The sender obtains the receiver public key PK_R from the approved key store and computes $W = RSA-OAEP(PK_R, K_s)$. Only the corresponding private key can recover K_s .

Step 5 - Secure envelope construction. The transmission envelope contains message ID, timestamp, sender/receiver identifiers, N , W , C , Tag , and versioned algorithm metadata. No

plaintext or unwrapped AES key is placed in the cloud envelope.

Step 6 - Cloud transport and optional storage. The envelope is sent through a TLS-protected channel to reduce exposure to network interception and downgrade attacks. The payload remains independently encrypted, so a cloud storage or routing compromise does not directly disclose plaintext. This defense-in-depth approach addresses the shared-infrastructure concerns highlighted in earlier studies [1], [3], [6].

Step 7 - Receiver validation. The receiver checks message freshness and the message ID against a replay cache, validates the sender and policy context, decrypts W with its RSA private key, and runs AES-GCM decryption. Any authentication-tag failure causes immediate rejection and an audit event.

Step 8 - Audit and retention. The system records the message ID, participants, timestamp, algorithm version, status, and integrity outcome. Plaintext and secret session keys are excluded from the log. For longer-term outsourced data, the audit layer can be extended with remote

possession or retrievability mechanisms developed by Ateniese et al. and Juels and Kaliski [7], [8], or privacy-preserving public auditing as studied by Wang et al. [9].

3. Prototype and Evaluation Plan

A reproducible prototype was implemented using the Python cryptography library. The runtime environment exposed an x86-64 Intel Xeon Platinum 8573C processor and approximately 5.8 GiB of memory. Three randomly generated payload sizes (1 MB, 5 MB, and 10 MB) were evaluated. Each AES-GCM encrypt/decrypt measurement was repeated 15 times and averaged. RSA-2048/OAEP session-key wrapping and unwrapping were averaged over 100 operations. To test integrity enforcement, 100 independently encrypted 4 KB ciphertexts were modified by flipping one bit before decryption; rejection by the GCM authentication check was recorded. These measurements evaluate cryptographic processing cost rather than Internet latency, which varies with network conditions and cloud region.

Table 2. Framework Security Controls

Security Need	Control	Protected Asset	Failure Response
Confidentiality	AES-256-GCM	Payload	No plaintext released
Key secrecy	RSA-2048/OAEP	Session key	Unwrap failure / reject
Integrity	GCM authentication tag	Ciphertext + associated data	Reject + audit
Replay resistance	Message ID + timestamp	Protocol state	Reject duplicate/stale
Authorization	Identity/policy check	Receiver access	Deny before decryption
Traceability	Security event log	Control plane	Record result without secrets

Results

The prototype results indicate that authenticated symmetric encryption contributes a small processing cost compared with typical wide-area cloud-transfer delays. Table 3 reports the mean

cryptographic time for the three tested payload sizes. Because the measurements were executed on one contemporary processor, the values should be interpreted as implementation-level evidence for feasibility rather than universal cloud-performance guarantees.

Table 3. Prototype Cryptographic Performance (mean values)

Payload	Encrypt (ms)	Decrypt (ms)	Enc. throughput	Dec. throughput	Envelope overhead*
1 MB	0.264	0.176	3787 MB/s	5686 MB/s	0.0271%
5 MB	1.800	1.553	2778 MB/s	3220 MB/s	0.0054%
10 MB	3.419	2.403	2925 MB/s	4161 MB/s	0.0027%

Session-Key and Integrity Tests

RSA-2048/OAEP wrapping of the 256-bit AES session key averaged approximately 0.028 ms, while private-key unwrapping averaged approximately 0.324 ms over 100 operations.

Since RSA is applied only to the small session key, its cost does not grow with payload size. The ciphertext itself adds a 16-byte authentication tag; including a 12-byte nonce and the 256-byte wrapped key produces a fixed 284-byte

cryptographic envelope before ordinary metadata. As payload size grows, this fixed overhead becomes proportionally smaller.

All 100 deliberately modified 4 KB ciphertexts were rejected by AES-GCM authentication. This result confirms that, in the tested implementation, a one-bit change in the protected ciphertext prevents plaintext acceptance. Such authenticated encryption provides a direct integrity check during transmission. It complements, rather than replaces, the remote-storage assurance goals of PDP, POR, and privacy-preserving auditing [7]-[9].

Security Coverage

The framework addresses passive interception because only ciphertext is visible on the cloud path; malicious modification because the GCM tag must verify; key exposure at the cloud provider because the data key is wrapped for the intended receiver; replay attempts because the receiver checks message identifiers and freshness; and unauthorized users because policy checks occur before key release or plaintext delivery. Even if co-resident workloads can observe shared infrastructure, the design reduces the value of exposed data because the provider is not given plaintext session keys, consistent with the risks discussed by Ristenpart et al. [3].

The principal limitation is endpoint trust. If a sender or receiver private key is compromised, cryptography cannot prevent misuse by that endpoint. Key revocation, protected private-key storage, certificate validation, and secure identity administration are therefore necessary operational controls. In addition, cryptographic protection does not prevent traffic analysis, availability attacks, or leakage through the application after decryption.

Conclusion and Discussion

The study presents a practical methodology for securing data transmission in cloud-based communication systems by combining multiple controls around the data itself. Earlier research established that cloud adoption expands security exposure through Internet-facing interfaces, outsourcing, multi-tenancy, dynamic infrastructure, and the reduced physical control of data [1]-[6], [11]. The proposed framework responds to those conditions by encrypting the payload before it enters the cloud domain, using a fresh symmetric key for each transmission, protecting that key with the authorized receiver's public key, authenticating the

ciphertext, checking policy and freshness information, and generating auditable security events. The performance experiment shows that this design is computationally practical on the tested environment. For 1 MB to 10 MB payloads, AES-256-GCM encryption and decryption completed in a few milliseconds or less, and RSA key wrapping added sub-millisecond cost. More importantly, every deliberately altered test ciphertext failed authentication, illustrating the value of authenticated encryption for preventing silent modification. The fixed cryptographic envelope also becomes negligible as message size grows. These measurements do not claim to represent total end-to-end cloud latency; instead, they isolate the processing introduced by the proposed security layer. The framework also creates clear extension points. Where organizations require attribute-level access policies, CP-ABE can move some authorization rules into the ciphertext itself [10]. Where encrypted files remain stored for long periods, provable data possession, proofs of retrievability, or privacy-preserving public auditing can provide stronger evidence that outsourced data remains intact [7]-[9]. These mechanisms are complementary: transmission encryption protects data while it is moving or temporarily stored, while remote-audit schemes address continued possession and integrity after outsourcing. A significant methodological advantage is separation of trust. The cloud provider can continue to supply routing, elasticity, storage, and communication services, but confidentiality does not depend solely on provider behavior. This is consistent with cryptographic cloud-storage concepts [4] and with PKI-based trust services proposed for cloud environments [5]. The approach therefore supports public and hybrid cloud deployments in which the organization needs technical controls that remain effective even when infrastructure is shared. Several limitations should guide future implementation. First, the prototype evaluates cryptographic processing and integrity rejection but does not model Internet congestion, packet loss, or distributed denial-of-service. Second, RSA-2048 and AES-256-GCM were selected as a conventional hybrid design; long-lived systems require algorithm agility so that cryptographic parameters can be replaced without redesigning the protocol. Third, replay protection depends on reliable message-state management, synchronized time policy, or durable identifiers. Fourth, endpoint compromise remains a dominant risk because authorized software necessarily processes plaintext. Secure key storage, certificate lifecycle management, least-privilege authorization, endpoint hardening, and

revocation procedures must therefore be part of a production deployment.

In conclusion, the methodology demonstrates that secure cloud communication can be constructed as an integrated pipeline rather than a single encryption step. Client-side authenticated encryption, protected session keys, policy enforcement, replay control, and auditability collectively provide a stronger security boundary for sensitive information. The design is compatible with security research available up to 2015 and offers a reproducible basis for implementation, benchmarking, and later extension to richer access-control and outsourced-integrity mechanisms.

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.
- M. Jensen, J. Schwenk, N. Gruschka, and L. Lo Iacono, "On Technical Security Issues in Cloud Computing," in 2009 IEEE International Conference on Cloud Computing, pp. 109–116, 2009. DOI: 10.1109/CLOUD.2009.60.
- T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," in Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 199–212, 2009. DOI: 10.1145/1653662.1653687.
- S. Kamara and K. E. Lauter, "Cryptographic Cloud Storage," in *Financial Cryptography and Data Security*, LNCS 6054, pp. 136–149, 2010. DOI: 10.1007/978-3-642-14992-4_13.
- 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," in 2012 International Conference on Computer Science and Electronics Engineering, vol. 1, pp. 647–651, 2012. DOI: 10.1109/ICCSEE.2012.193.
- G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," in Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 598–609, 2007. DOI: 10.1145/1315245.1315318.
- Juels and B. S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," in Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 584–597, 2007. DOI: 10.1145/1315245.1315317.
- 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.
- J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in 2007 IEEE Symposium on Security and Privacy, pp. 321–334, 2007. DOI: 10.1109/SP.2007.11.
- M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015. DOI: 10.1016/j.ins.2015.01.025.