Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

A Proposed Methodology to Improve the Integrity and Confidentiality of Cloud Computing Services

Mohammad Yaser Hashisho
Department of Computer Systems and Networks
Faculty of Informatics Engineering
University of Aleppo, Syria

Souheil Khawatmi
Department of Computer Systems and Networks
Faculty of Informatics Engineering
University of Aleppo, Syria

***Abstract* - Cloud storage has to keep outsourced data confidential and intact without giving up the processing speed and the low resource use that make the cloud attractive. Symmetric ciphers used alone are fast but leave the secret key exposed while it is shared, and schemes that add a blockchain ledger to guarantee integrity pay for it with long delays and heavy use of server resources. This paper proposes an authenticated hybrid cryptosystem for data at rest that encrypts file chunks with the Advanced Encryption Standard, protects the chunk keys with elliptic curve cryptography on the prime256v1 curve, and verifies every chunk with a hash based message authentication code, while the encrypted keys are isolated in a separate key vault. The methodology was implemented in the CloudSim simulator and compared with single symmetric ciphers, with a recent hybrid that pairs the same two algorithms with a plain hash, with a blockchain based hybrid, and with a hybrid built on the Rivest Shamir Adleman algorithm. It encrypted 40 percent faster than the hash based hybrid. Against the blockchain based hybrid it reduced encryption time by up to 89 percent and decryption time by up to 70 percent for files of 30 to 100 megabytes, lowered power draw by 15 to 29 percent, and kept the avalanche effect above 50 percent. For payloads of 64 to 288 bytes encryption took a constant 3.5 milliseconds, more than 99 percent below the compared hybrid. The results show that integrity can be verified in the cloud without a blockchain at a small and predictable cost.**

Keywords - cloud computing; hybrid encryption; elliptic curve cryptography; prime256v1; AES; HMAC; avalanche effect; CloudSim

I. INTRODUCTION

Cloud computing delivers software, platforms and infrastructure as services over the Internet, so that organizations and individuals can store, process and share data from any place without owning the underlying hardware [1]. Storage is the most widely used of these services: it acts both as a large backup and as a primary medium, and it removes the need to keep data on a personal device. The same property is the main obstacle to adoption. Data placed on servers that the owner does not control can be read, sold or modified by intruders, and in some cases by the administrators of the provider, and the surveys of cloud security list data protection ahead of every other challenge [2].

The technical problem has three parts. First, sensitive data on shared external storage is exposed to theft and to unauthorized modification. Second, a symmetric cipher such as the Advanced Encryption Standard (AES) encrypts large volumes quickly, but used alone it offers no safe way to distribute the secret key between the user and the cloud server. Third, a conventional asymmetric cipher such as Rivest-Shamir-Adleman (RSA) solves key distribution only with keys of 2048 or 3072 bits, which makes it slow and limits it to very small blocks. Recent proposals close the integrity gap by recording data fingerprints on a blockchain, but the consensus and ledger operations add a delay and a resource load that real-time storage cannot afford.

This paper proposes a methodology that secures data at rest without that load. Each file is divided into independent chunks; every chunk is encrypted with AES under its own key and authenticated with a hash-based message authentication code (HMAC); the per-chunk keys are encrypted with elliptic curve cryptography (ECC) on the prime256v1 curve and kept in a key vault that is separate from the data servers. The contributions are:

- an authenticated hybrid design (AES, ECC prime256v1 and HMAC-SHA256) in which ciphertext, tags and keys are stored apart, so that a breach of one store does not expose the data;
- an implementation in CloudSim with an evaluation of execution time, power and energy, throughput, storage overhead and avalanche effect on files of 30 to 100 MB and packets of 64 to 288 bytes;
- a comparison with single symmetric ciphers, with an ECC-AES-SHA hybrid, with a blockchain-enabled AES-ECC hybrid and with an RSA-AES hybrid, together with recommended chunking thresholds and the scope of suitable applications.

Section II reviews related work. Section III describes the proposed methodology and Section IV the experimental setup. Section V presents and discusses the results, and Section VI concludes the paper.

II. RELATED WORK

A. Security of Data Stored in the Cloud

Saroj et al. [3] used threshold cryptography: the data owner divides users into groups, gives each group one decryption key and each member only a share of it, and controls access through a capability list, while a one-time session key agreed with Diffie-Hellman protects the transfer. Harikrishna and Amuthan [4] described key generation, digital signature and signature verification with an access list and Diffie-Hellman key exchange; the approach remains exposed to denial-of-service attacks. Chandu and Kata [5] combined block and stream ciphers for confidentiality with hashing for integrity, but

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

did not address user verification. Ziglari and Yahya [6] placed an encryption service between data processing and storage and hid the key from both; the repeated transfers between services are the weakness of the model. In [7] all files are merged into one container that is encrypted, fragmented and uploaded with a protected storage key, which forces a user to download and decrypt the whole container to recover a single file.

Singh and Singh [8] encrypted with AES under a key derived from the user password with HMAC and authenticated the user with a digital signature, yet did not achieve data integrity. Rewagad and Pawar [9] combined a hash-based digital signature, Diffie-Hellman key exchange and AES, a three-way protection that keeps speed acceptable. The survey in [10] covers device-based identification, attribute-based encryption and blockchain for privacy, and notes that blockchain needs very large resources.

B. Selection of the Cryptographic Algorithms

Table I summarizes the studies that guided the choice of algorithms. The reviews in [11] and [12] found AES more secure and faster than DES, 3DES, Twofish and RSA, and the survey of cryptanalytic attacks in [13] shows that brute-force, ciphertext-only, side-channel, boomerang, power-analysis and algebraic attacks remain theoretical or implementation-dependent against the full cipher. Selvi and Sakthivel [14] built a hybrid of ECC, AES and SHA and showed that it gives stronger encryption with less computing power and bandwidth than 3DES and AES; the size of an ECC key grows linearly with the equivalent symmetric key, while an RSA key grows far faster. Their scheme, however, fixes no standard curve parameter, verifies integrity with an unkeyed hash and does not plan how files are partitioned. Bakro et al. [15] combined AES and ECC with a blockchain and reported a high level of security, at the price of the execution time and resources that the ledger consumes; the file is also split into two parts, which loads memory and the network. Akter et al. [16] encrypted the plaintext with 128-bit AES and the AES key with RSA; a 2048-bit RSA key can only encrypt a few hundred bytes and the operation is slow.

TABLE I. STUDIES USED TO SELECT THE ALGORITHMS

Ref.	Algorithms	Contribution	Limitation
[11]	AES, DES, RSA	Analytical comparison to identify the best cipher for protecting information	Theoretical study with no implementation in a cloud environment
[12]	Enhanced AES (14 rounds), DES, 3DES, Twofish	More rounds to raise security and reduce processing time	Comparison limited to small files
[13]	AES (cryptanalysis)	Survey of side-channel, brute-force and algebraic attacks on AES	Analytical survey, no proposed model
[14]	ECC, AES, SHA	Equivalent security with smaller keys and lower energy	No standard curve parameter; keys not protected in an ideal way
[15]	AES, ECC, blockchain	High security level for the cloud storage infrastructure	Large increase in execution time and resource use caused by the blockchain
[16]	AES, RSA	Plaintext encrypted with AES and the AES key with RSA	RSA is slow and limited to a few hundred bytes

Two gaps follow from this review. Hybrids that protect the key with RSA inherit its cost, and hybrids that rely on a blockchain for integrity are too slow for real-time storage. The methodology below keeps the AES-ECC combination of [14] and [15], fixes the curve to a standard parameter, and replaces the plain hash and the ledger with keyed authentication tags and isolated key storage.

III. PROPOSED METHODOLOGY

A. Design Overview

The methodology targets the three security goals of confidentiality, integrity and authenticated key exchange with one standardized primitive for each. AES provides confidentiality of the stored chunks. HMAC with SHA-256 provides integrity and detects any modification of a chunk. ECC on the prime256v1 curve protects and exchanges the keys; it was preferred to RSA because it gives equivalent security with far smaller keys and lighter operations [14], [17]. The design applies defense in depth: a file is split into chunks that are protected independently, and ciphertext, authentication tags and keys are held in three different places. Fig. 1 shows the storage and retrieval phases.

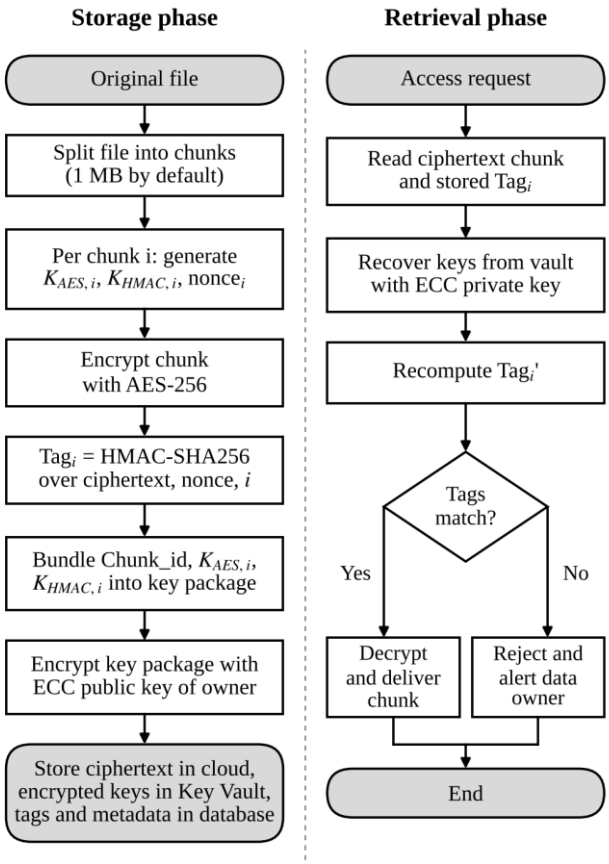


Fig. 1. Storage and retrieval phases of the proposed methodology.

B. Storage Phase

The file is divided into chunks of fixed size (1 MB by default). For chunk i the client generates a random 256-bit encryption key $K_{AES,i}$, a random 256-bit authentication key $K_{HMAC,i}$ and a 128-bit nonce. The chunk is encrypted with AES

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

in cipher block chaining mode, and a tag is computed over the ciphertext, the nonce and the chunk index:

$$Tag_i = \text{HMAC-SHA256}(K_{HMAC,i},\ C_i \parallel Nonce_i \parallel i) \quad (1)$$

The original chunk is then replaced by its ciphertext. The keys of the chunk are bundled into a key package,

$$KP_i = (Chunk_id,\ K_{AES,i},\ K_{HMAC,i},\ Nonce_i) \quad (2)$$

which is encrypted with the ECC public key of the data owner. The wrapping key is obtained from an elliptic curve Diffie-Hellman exchange on prime256v1 followed by HKDF-SHA256, so no secret key ever travels over the channel. The ciphertext chunks are uploaded to the cloud store, the encrypted key packages are written to the key vault and the tags are kept in a database for later verification. Fig. 2 lists the procedure.

Input: file F, owner public key PK; chunk size s = 1 MB
1: $chunks \leftarrow$ Split(F, s); $n \leftarrow$ number of chunks
2: for i = 1 to n do
  3: $K_{AES,i} \leftarrow$ Random(256 bits); $K_{HMAC,i} \leftarrow$ Random(256 bits)
  4: $Nonce_i \leftarrow$ Random(128 bits)
  5: $C_i \leftarrow$ AES-Encrypt($chunks[i]$, $K_{AES,i}$, $Nonce_i$)
  6: $Tag_i \leftarrow$ HMAC-SHA256($K_{HMAC,i}$, $C_i \parallel Nonce_i \parallel i$)
  7: $KP_i \leftarrow$ ($Chunk_id$, $K_{AES,i}$, $K_{HMAC,i}$, $Nonce_i$, timestamp)
  8: $EKP_i \leftarrow$ ECC-Encrypt(KP_i, PK)
  9: $Meta_i \leftarrow$ (file ID, i, n, timestamp)
10: end for
11: Upload $\{C_i\}$ to cloud storage; store $\{Tag_i, Meta_i\}$ in the tag database
12: Store $\{EKP_i\}$ and the package metadata in the key vault

Fig. 2. Pseudocode of the storage procedure.

C. Retrieval and Verification

On an access request the stored ciphertext and its tag are read, the key package is recovered from the vault with the ECC private key of the owner, and the tag is recomputed. If the two tags are equal the chunk is decrypted and delivered; otherwise the request is rejected and the owner is alerted. Retrieval needs no new random numbers and no new ephemeral keys: a single scalar multiplication on the curve recovers the AES key, after which the blocks are decrypted directly.

D. Key Vault and Package Metadata

The key vault is a dedicated secure store for secret keys and certificates, separate from the general database. It holds only the ECC-encrypted key packages, so an attacker who breaks into the vault still needs the private key, which never leaves the data owner. An attacker who breaks into the main storage server obtains ciphertext that cannot be decrypted. Tags are stored apart from the chunks for the same reason: a tag kept beside its chunk would let an attacker replace the chunk and recompute a matching tag. Each package carries metadata: package identifier, identifier of the original file, number of chunks, version of the cipher suite (AES-256 with ECC-256), creation date and state (active or invalid).

E. Compliance With Standards

Every component follows a published standard, as listed in Table II. AES in cipher block chaining mode follows NIST SP 800-38A [18]; key establishment by elliptic curve Diffie-Hellman follows NIST SP 800-56A [19]; the use of HMAC follows NIST SP 800-107 [20] and its construction RFC 2104 [21]; and AES is also an international standard under ISO/IEC 18033-3 [22]. Neither AES nor ECC alone satisfies all of them, whereas the hybrid does, at a 128-bit security level.

TABLE II. STANDARDS SATISFIED BY THE HYBRID AND BY ITS COMPONENTS

Standard	Scope	Hybrid	AES	ECC
NIST SP 800-38A [18]	AES-CBC mode of operation	Yes	Yes	No
NIST SP 800-56A [19]	ECC key establishment	Yes	No	Yes
RFC 2104 [21]	HMAC construction	Yes	No	No
ISO/IEC 18033-3 [22]	International AES standard	Yes	Yes	No

IV. EXPERIMENTAL SETUP

The methodology was implemented in Java inside CloudSim, a widely used simulator that models data centers, processors, memory, network and storage and reports the results of each run [23]. The AES and ECC implementations were both integrated in Java because the simulator does not support components written in different languages. Table III gives the simulation parameters.

TABLE III. SIMULATION PARAMETERS

Parameter	Value
Simulator	CloudSim 3.0.3
Programming language	Java (JDK 11)
Elliptic curve	prime256v1
AES key size	128, 192 and 256 bits
ECC key size	256 bits
Integrity function	HMAC-SHA256
Cloud files	1 MB to 1000 MB
Small payloads	64 to 288 bytes
Host	Windows 10, Core i5, 8 GB RAM, 256 GB SSD and 1 TB hard disk

Three comparisons were made: with the ECC-AES-SHA hybrid of [14] and the single ciphers AES, 3DES and Blowfish reported with it, at a 256-bit key on a 5 MB file; with the blockchain-enabled AES-ECC hybrid of [15] on files of 30, 60 and 100 MB; and with the RSA-AES hybrid of [16] on payloads of 64 to 288 bytes. The scheme of [14] checks integrity with plain SHA-256, uses a generic curve with no accelerated standard parameter and runs as a web application inside the simulator; the system of [15] likewise uses a generic 256-bit curve and a web application connected to a local server. The metrics are encryption and decryption time, power consumption, throughput, size of the encrypted file and avalanche effect. The avalanche effect is the share of ciphertext bits that change when one bit of the key or of the plaintext is flipped,

$$AE = (N_{flipped} / N_{total}) \times 100\% \quad (3)$$

with an ideal value of 50% and an accepted range of 45% to 55%. Throughput is the amount of data processed per unit of time,

$$TP = S / T \quad (4)$$

where S is the data size in MB and T the execution time in seconds. The energy of one operation is obtained from the measured average power P and the execution time as $E = P \times T$.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

V. RESULTS AND DISCUSSION

A. Comparison With the ECC-AES-SHA Hybrid and Single Ciphers

Table IV and Fig. 3 compare the proposed hybrid with the ECC-AES-SHA hybrid of [14] and with AES, 3DES and Blowfish at a 256-bit key. The proposed methodology completes encryption, authentication and key exchange in 4.99 ms, against 8.33 ms for the hybrid of [14]: a saving of 3.34 ms, or 40.10%, which is a speed-up of about 1.67 times. It also draws 3.95 W instead of 4.15 W. The gain comes from the accelerated standard curve prime256v1 and from processing independent chunks in memory, whereas [14] relies on a generic, unoptimized curve equation. Compared with AES alone, the elliptic curve and authentication layers add about 2 ms and 0.87 W, and the hybrid remains faster than 3DES and lighter than 3DES and Blowfish. This is the price of removing the key-sharing weakness of a symmetric cipher used alone.

TABLE IV. ENCRYPTION TIME AND POWER CONSUMPTION AT A 256-BIT KEY (5 MB FILE)

Scheme	Encryption time (ms)	Power (W)
Proposed hybrid	4.99	3.95
ECC-AES-SHA hybrid [14]	8.33	4.15
AES	3.00	3.08
3DES	8.00	4.65
Blowfish	4.00	4.25

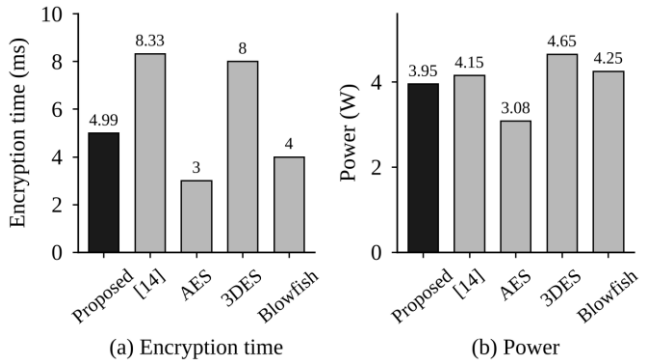


Fig. 3. Encryption time and power of the proposed hybrid, the ECC-AES-SHA hybrid and single ciphers at a 256-bit key.

Table V gives the avalanche effect in a 128-bit block. A one-bit change of the key flips 67 bits (52.34%) in the proposed hybrid and 63 bits (49.22%) in [14]; a one-bit change of the plaintext flips 65 bits (50.78%) against 62 bits (48.44%). The higher key sensitivity follows from scalar multiplication on the standard curve combined with an independent key for every chunk, where [14] derives its key directly through SHA-256.

TABLE V. AVALANCHE EFFECT COMPARED WITH THE ECC-AES-SHA HYBRID [14]

Test	Hybrid	[14]
1-bit key change: bits flipped	67 of 128	63 of 128
1-bit key change: avalanche effect	52.34%	49.22%
1-bit plaintext change: bits flipped	65 of 128	62 of 128
1-bit plaintext change: avalanche effect	50.78%	48.44%

B. Comparison With the Blockchain-Enabled Hybrid

Table VI compares the proposed methodology with the AES-ECC-blockchain system of [15]. The reference system carries a nearly constant penalty of about 1000 ms at every file size, the cost of the consensus and ledger operations. The proposed system encrypts a 30 MB file in 120 ms instead of 1130 ms, a reduction of 89.38%; the reduction is 80.95% at 60 MB and 71.43% at 100 MB (Fig. 4). Decryption is 70%, 66% and 68.89% faster at the three sizes, about two thirds on average, because the reference system must query the ledger and validate the fingerprints and the smart-contract permissions before any block is released. In the proposed system decryption is also 25% to 30% faster than encryption of the same file, since no random values or ephemeral keys are generated on retrieval.

TABLE VI. PERFORMANCE COMPARED WITH THE BLOCKCHAIN-ENABLED HYBRID [15]

Metric	30 MB		60 MB		100 MB	
	Hybrid	[15]	Hybrid	[15]	Hybrid	[15]
Encryption time (ms)	120	1130	240	1260	400	1400
Decryption time (ms)	90	300	170	500	280	900
Power (W)	17	24	26	35	40	47
Energy (J)[a]	2.04	27.12	6.24	44.10	16.00	65.80
Storage overhead (KB)	4	9	8	13	13	18

[a] Derived as average power multiplied by encryption time.

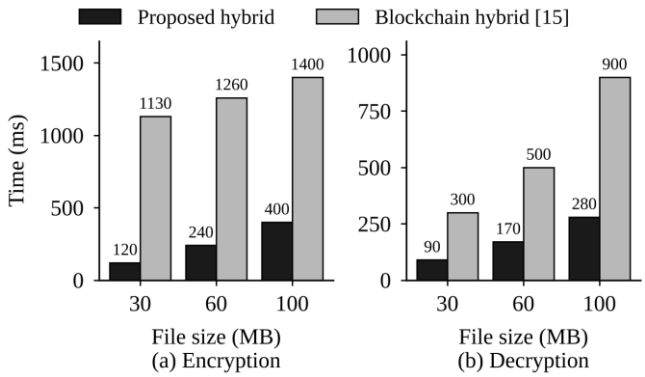


Fig. 4. Encryption and decryption time compared with the blockchain-enabled hybrid.

The reference system draws about 7.67 W more on average, so the proposed system saves 29.17%, 25.71% and 14.89% of the power at 30, 60 and 100 MB. Because it also finishes much sooner, the energy spent per encryption falls by more than 75% at every size (Fig. 5), which suits green cloud computing and

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

battery-powered edge devices: the processor and the network interface are freed from mining and ledger synchronization.

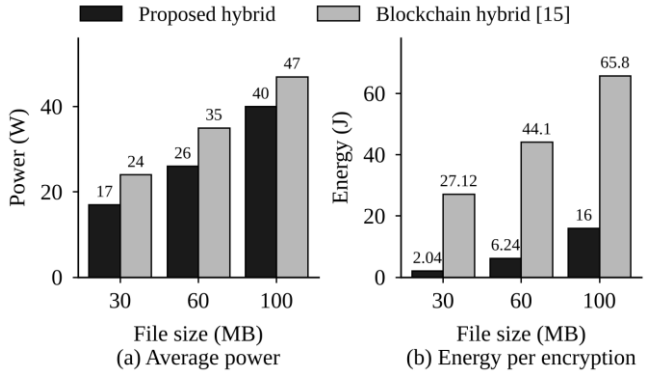


Fig. 5. Average power and energy per encryption compared with the blockchain-enabled hybrid.

The encrypted file grows by 4, 8 and 13 KB for the three sizes, against 9, 13 and 18 KB in the reference system (Fig. 6). Removing the mandatory block headers, about 5 KB per file, cuts the added storage by 55.56%, 38.46% and 27.78%.

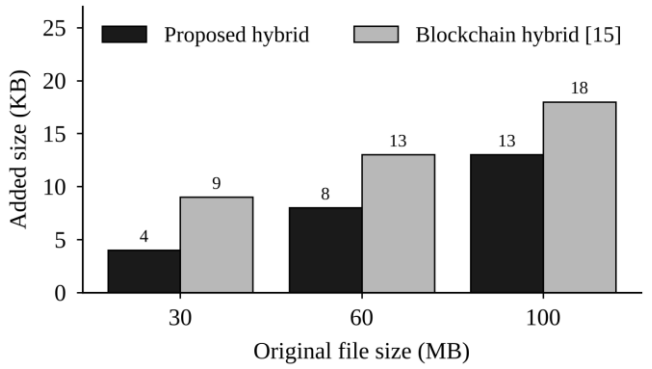


Fig. 6. Size added to the encrypted file compared with the blockchain-enabled hybrid.

Table VII collects the remaining indicators. Cryptographic throughput reaches 250 MB/s for encryption and 350 MB/s for decryption, 13.64% and 9.38% above the 220 MB/s and 320 MB/s of the cryptographic stage of the reference system, taken without its ledger operations. The avalanche effect is 52.34% for a one-bit key change and 50.78% for a one-bit plaintext change in a 128-bit block, against 50.78% and 49.22%. A one-bit change in the secret scalar moves the resulting curve point completely, and the key derivation function and the AES key expansion then spread the change over more than half of the ciphertext bits. Leaving out the blockchain therefore takes nothing from the statistical strength of the ciphertext against linear and differential cryptanalysis. Integrity is verified in about 0.05 ms, against 200 to 500 ms through the ledger, and verification works without a network connection. The blockchain keeps one advantage: it prevents tampering by the cloud provider itself, which keyed tags alone do not.

TABLE VII. FURTHER INDICATORS COMPARED WITH THE BLOCKCHAIN-ENABLED HYBRID [15]

Indicator	Hybrid	[15]
Avalanche effect, 1-bit key change	52.34% (67 of 128 bits)	50.78% (65 of 128 bits)
Avalanche effect, 1-bit plaintext change	50.78% (65 of 128 bits)	49.22% (63 of 128 bits)
Cryptographic throughput, encryption	250 MB/s	220 MB/s
Cryptographic throughput, decryption	350 MB/s	320 MB/s
Integrity verification time	0.05 ms	200 to 500 ms
Detects modification	Yes	Yes
Prevents tampering by the provider	No	Yes
Additional storage cost	Very low	Very high
Suitable for large files	Yes	Limited by block size
Verification without network access	Yes	No

C. Comparison With the RSA-AES Hybrid

Table VIII and Fig. 7 compare the methodology with the RSA-AES hybrid of [16] on small payloads, the typical load of Internet of Things devices. Encryption takes 3.5 ms instead of about 1047 ms, an improvement of 99.66%, and decryption 1.2 ms instead of about 54 ms, an improvement of 97.78%. Throughput is roughly 300 times higher at every size. The time of the proposed system does not change with the payload because AES processes 16-byte blocks: 64 bytes are 4 blocks and 288 bytes are 18 blocks, which take microseconds. The measured 3.5 ms is in practice the fixed cost of establishing the session and deriving the shared key by one scalar multiplication on prime256v1.

TABLE VIII. RESULTS FOR SMALL PAYLOADS COMPARED WITH THE RSA-AES HYBRID [16]

Size	Encryption (ms)		Decryption (ms)		Throughput (KB/s)	
(bytes)	Hybrid	[16]	Hybrid	[16]	Hybrid	[16]
64	3.5	1047.07	1.2	53.90	17.4	0.058
96	3.5	1047.08	1.2	53.97	26.1	0.087
144	3.5	1047.10	1.2	53.99	39.1	0.131
176	3.5	1047.20	1.2	53.10	47.8	0.160
208	3.5	1047.30	1.2	53.18	56.3	0.189
240	3.5	1047.40	1.2	53.25	65.0	0.218
288	3.5	1047.50	1.2	53.44	77.9	0.262

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

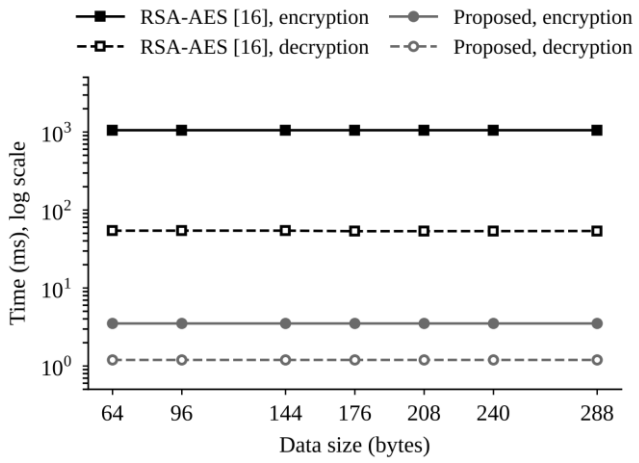


Fig. 7. Encryption and decryption time for small payloads compared with the RSA-AES hybrid.

D. Security Analysis

Table IX lists the attacks on files stored in the cloud that each scheme withstands. The hybrid covers unauthorized access, storage misconfiguration, hash collision, unauthorized modification, side-channel and cloud man-in-the-middle attacks, a set that neither AES nor ECC covers in full alone, and it resists replay under conditions. Two cases remain open: recovery of deleted data and a malicious internal administrator, both of which the blockchain-based system handles.

TABLE IX. ATTACKS WITHSTOOD BY EACH SCHEME

Attack	Hybrid	AES	ECC	[15]
Unauthorized access	Yes	Yes	No	Yes
Recovery of deleted data	No	No	No	Yes
Storage misconfiguration	Yes	Yes	No	Yes
Malicious internal administrator	No	No	No	Yes
Replay	Partial	Partial	No	No
Hash collision	Yes	No	No	Yes
Unauthorized modification	Yes	Yes	No	Yes
Side channel	Yes	No	Yes	Yes
Cloud man-in-the-middle	Yes	Yes	No	Yes

E. Recommended Thresholds and Scope

Chunking multiplies the number of keys and tags, so its parameters must be bounded. A chunk should not be smaller than 1 MB, a file should not be split into more than 10 000 chunks, and files above 10 GB need additional optimization of key generation. Table X gives the recommended settings by file size.

TABLE X. RECOMMENDED CHUNKING BY FILE SIZE

File size	Split	Chunk size	Chunks	Assessment
Below 1 MB	No	Whole file	1	Avoids splitting cost
1 MB to 100 MB	Yes	1 MB	1 to 100	Ideal
100 MB to 1 GB	Yes	5 MB	20 to 200	Good
1 GB to 10 GB	Yes	10 MB	100 to 1000	Acceptable

With a constant 3.5 ms for small packets and 250 MB/s for large files, the methodology suits electronic health records and medical images, real-time financial and digital banking systems, confidential government documents, personal data on mobile phones, and cloud-connected Internet of Things devices with limited resources. It is not intended for hard real-time applications such as autonomous driving, robotic surgery, competitive online gaming and drones, nor for decentralized trustless environments that require non-repudiation.

VI. CONCLUSION

This paper presented an authenticated hybrid methodology for cloud data at rest that combines AES for confidentiality, ECC on prime256v1 for key protection and HMAC-SHA256 for integrity, with chunk-level isolation and a key vault separated from the data servers. In simulation it encrypted 40.10% faster than a recent ECC-AES-SHA hybrid and added about 2 ms to AES alone while removing the key-sharing weakness; it cut encryption time by up to 89.38%, decryption time by up to 70% and energy by more than 75% relative to a blockchain-enabled hybrid; it reduced the added storage by 27.78% to 55.56%; and it encrypted small payloads more than 99% faster than an RSA-AES hybrid, with an avalanche effect above 50% throughout. Integrity of cloud data can thus be verified in a fraction of a millisecond without the cost of a ledger.

The methodology has limits. It costs more time and power than a symmetric cipher alone, and it does not protect against a malicious internal administrator or the recovery of deleted data. Future work will combine the design with post-quantum key encapsulation such as ML-KEM alongside prime256v1, study search and indexing over encrypted data with homomorphic encryption, and offload the curve multiplications and AES rounds to graphics processors and FPGAs for very large data sets.

REFERENCES

[1] F. S. Al-Anzi, S. K. Yadav, and J. Soni, “Cloud computing: Security model comprising governance, risk management and compliance,” Indira Gandhi Delhi Technical University, Delhi, India, 2015.

[2] F. Shahzad, “State-of-the-art survey on cloud computing security challenges, approaches and solutions,” in Proc. 6th Int. Symp. Applications of Ad hoc and Sensor Networks, 2014, pp. 357–362.

[3] S. K. Saroj, S. K. Chauhan, A. K. Sharma, and S. Vats, “Threshold cryptography based data security in cloud computing,” in Proc. IEEE Int. Conf. Computational Intelligence and Communication Technology, 2015.

[4] P. Harikrishna and A. Amuthan, “A survey of testing as a service in cloud computing,” in Proc. Int. Conf. Computer Communication and Informatics, Coimbatore, India, Jan. 2016.

[5] P. M. S. S. Chandu and D. Kata, “Integrating and enhancing the quality of services in cloud computing with software testing,” Sathyabama University, Chennai, India, 2016.

[6] H. Ziglari and S. Yahya, “Deployment models: Enhancing security in cloud computing environment,” in Proc. 22nd Asia-Pacific Conf. Communications (APCC), 2016.

[7] R. Salloum, “A proposed study to improve the security of data stored in cloud computing,” M.S. thesis, Univ. of Aleppo, Aleppo, Syria, 2017 (in Arabic).

[8] N. Singh and N. K. Singh, “Information security in cloud computing using encryption techniques,” Int. J. Scientific and Engineering Research, vol. 5, no. 4, 2019.

[9] P. Rewagad and Y. Pawar, “Use of digital signature with Diffie Hellman key exchange and AES encryption algorithm to enhance data security in cloud computing,” in Proc. Int. Conf. Communication Systems and Network Technologies (CSNT), 2013, pp. 437–439.

[10] A. I. Tahirkheli et al., “A survey on modern cloud computing security over smart city networks: Threats, vulnerabilities, consequences,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

countermeasures, and challenges," Electronics, vol. 10, no. 15, art. 1811, 2021.

[11] R. Sood and H. Kaur, "A literature review on RSA, DES and AES encryption algorithms," in Emerging Trends in Engineering and Management, V. Dhiman and P. Dhand, Eds. India: SCRS, 2023, pp. 57–63.

[12] N. A. M. Ariffin and A. Y. A. Ashawesh, "Enhanced AES algorithm based on 14 rounds in securing data and minimizing processing time," J. Physics: Conf. Series, vol. 1793, art. 012066, 2021.

[13] H. Zodpe and A. Shaikh, "A survey on various cryptanalytic attacks on the AES algorithm," Int. J. Next-Generation Computing, vol. 12, no. 2, pp. 115–123, April 2021.

[14] P. Selvi and S. Sakthivel, "A hybrid ECC-AES encryption framework for secure and efficient cloud-based data protection," Scientific Reports, vol. 15, art. 30867, 2025.

[15] M. Bakro, S. K. Bisoy, A. K. Patel, and M. A. Naal, "Hybrid blockchain-enabled security in cloud storage infrastructure using ECC and AES algorithms," in Blockchain Based Internet of Things. Singapore: Springer, 2022.

[16] R. Akter, M. A. R. Khan, F. Rahman, S. J. Soheli, and N. J. Suha, "RSA and AES based hybrid encryption technique for enhancing data security in cloud computing," Int. J. Computational and Applied Mathematics and Computer Science, vol. 3, pp. 60–71, 2023.

[17] J. Joy and S. Devaraju, "Securing cloud data through efficient using ECC algorithm," Educational Administration: Theory and Practice, vol. 30, no. 5, pp. 9421–9429, 2024.

[18] M. Dworkin, "Recommendation for block cipher modes of operation: Methods and techniques," NIST Special Publication 800-38A, December 2001.

[19] E. Barker, L. Chen, A. Roginsky, A. Vassilev, and R. Davis, "Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography," NIST Special Publication 800-56A, Rev. 3, April 2018.

[20] Q. Dang, "Recommendation for applications using approved hash algorithms," NIST Special Publication 800-107, Rev. 1, August 2012.

[21] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," IETF RFC 2104, February 1997.

[22] Information Technology - Security Techniques - Encryption Algorithms - Part 3: Block Ciphers, ISO/IEC 18033-3, 2010.

[23] K. Bahwaireth, L. Tawalbeh, E. Benkhelifa, Y. Jararweh, and M. A. Tawalbeh, "Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications," EURASIP J. Information Security, vol. 2016, art. 15, 2016.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)