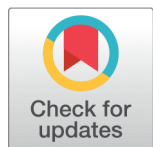


ORIGINAL ARTICLE



OPEN ACCESS

Received: 23/07/2026

Accepted: 01/08/2026

Published: 29/08/2026

Citation: Fathimamary B, Remy J, Geetha R (2026) A Novel Three-Layered Security Framework for Enhancing Data Confidentiality in Cloud Storage Environments. Indian Journal of Science and Technology 19(31): 2197-2205. <https://doi.org/10.17485/IJST/V19I31.1093>

* **Corresponding author.**

fathimamary_cc2@mail.sjctni.edu

Funding: None

Competing Interests: None

Copyright: © 2026 Fathimamary et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Published By Indian Society for Education and Environment ([iSee](#))

ISSN

Print: 0974-6846

Electronic: 0974-5645

A Novel Three-Layered Security Framework for Enhancing Data Confidentiality in Cloud Storage Environments

B Fathimamary^{1*}, J Remy², R Geetha³

¹ Department of Commerce Computer Applications, St.Joseph's College (Autonomous), Tiruchirappalli-620 002, Tamilnadu, India

² Department of Computer Science, St.Xavier's College (Autonomous), Palayamkottai, Tirunelveli-627 002, Tamilnadu, India

³ Department of Computer Applications, St.Xavier's College(Autonomous), Palayamkottai, Tirunelveli-627 002, Tamilnadu, India

Abstract

Objectives: Cloud data confidentiality is conventionally achieved by increasing the computational strength of a single encryption algorithm, which raises processing time and ciphertext size in proportion to security level. This study aims to achieve high confidentiality with low computational and storage overhead by decoupling security strength from a single heavy cipher and instead distributing protection across three lightweight, sequential transformations. **Method:** A three-layer framework is proposed, combining Word-based Magic Rectangle Alphanumeric Data Obfuscation (WMRADO), deterministic matrix-based scrambling, and single-key binary XOR encryption. Plaintext is first converted into a compressed alphanumeric code using a seed-driven 8×8 substitution square, then structurally disordered through positional matrix scrambling, and finally masked with a bitwise XOR operation before upload. The framework was implemented in Java (JDK 8) on an Intel Core i7-8700K workstation with 16 GB DDR4 RAM running Windows 10, and benchmarked against DES, AES, AES-Blowfish and AES-RSA using four plaintext sizes (112 B, 2,305 B, 7,894 B and 153,422 B) across ciphertext size, encryption time, decryption time and projected Google Cloud Storage cost. **Findings:** The proposed framework produced ciphertext 3–18 times smaller than DES/AES and up to 99.97% smaller than AES-RSA at the largest tested size (43 bytes versus 168,302 bytes). Encryption time was reduced by 9.5–52.0% relative to DES/AES and by up to 77.4% relative to AES-RSA, while decryption time was reduced by 3.0% relative to DES/AES and by up to 97.9% relative to AES-Blowfish/AES-RSA at large data sizes. Projected monthly storage cost fell by 99.5–99.8% relative to unencrypted storage and remained below both MRADO- and Moncrypt-obfuscated baselines at every tested file size. **Novelty:** The framework is the first to combine word-level alphanumeric obfuscation, matrix scrambling and XOR masking into a single seed/key pipeline evaluated jointly on security, latency and storage cost, rather than treating obfuscation and encryption, or security and efficiency, as separate design problems.

Keywords: cloud computing; cloud storage security; data confidentiality; data obfuscation; layered encryption; WMRADO; matrix scrambling; XOR encryption

1 Introduction

Cloud storage has become core infrastructure for enterprises, healthcare providers, financial institutions and government agencies that require scalable, low-maintenance data hosting⁽¹⁾. Outsourcing data to third-party infrastructure, however, exposes organisations to unauthorised access, insider threats and network-based attacks, which makes confidentiality — alongside integrity and availability — one of the three pillars of information security that a cloud tenant cannot delegate away^{(2), (3)}.

Symmetric algorithms such as DES, AES and Blowfish protect large data volumes efficiently with a single shared key, whereas asymmetric schemes such as RSA offer stronger key-distribution guarantees at substantially higher computational cost⁽⁴⁾. Hybrid constructions that combine both families (e.g., AES-RSA, AES-Blowfish) improve resistance to key-recovery attacks but inherit the overhead of their asymmetric component, and none of these single- or dual-mechanism schemes prevents full recovery of the plaintext once a single key is compromised⁽⁵⁾.

To address this, recent work has explored multi-layer and triple-encryption architectures. Kulshrestha et al.⁽⁶⁾ combined three probabilistic ciphers to raise the cost of brute-force recovery, and Ramachandra et al.⁽⁷⁾ applied triple-pass DES to large-scale cloud datasets; both report stronger security at the cost of increased processing time and larger ciphertext. Obfuscation-based methods provide a lighter-weight first line of defence: magic-square and related alphanumeric transformations conceal plaintext structure ahead of encryption while adding little computational cost^{(8), (9)}. More recent studies combine encoding with layered protection — Mishra et al.⁽¹⁰⁾ proposed a multi-layer encoding framework for cloud privacy, and Vinnarasi and Amalarethinam⁽¹¹⁾ introduced a Secured Layered Technique (SLT-DSA) stacking several security primitives — yet both still rely on a conventional block cipher for at least one layer, which limits achievable gains in ciphertext size and latency. Complementary directions include fine-grained access control⁽¹²⁾ and quantum-resistant hybrid schemes⁽¹³⁾, which strengthen authorisation and long-term resilience respectively without addressing ciphertext size or per-operation latency, together with end-to-end symmetric pipelines built around commercial cloud APIs⁽¹⁴⁾ and multi-factor, steganography-assisted protection⁽¹⁵⁾, both of which add procedural rather than computational overhead.

Across this literature, four gaps recur. First, obfuscation and encryption are mostly studied as independent problems, so the compounded confidentiality benefit of combining them within a single pipeline is rarely evaluated^{(9), (16)}. Second, many state-of-the-art schemes buy security with additional computation, a cost that scales poorly with the large object sizes typical of cloud storage^{(17), (18)}. Third, conventional encryption schemes enlarge ciphertext relative to plaintext, adding directly to commercial storage cost, and a lightweight alternative that avoids this penalty without weakening security has yet to be established^{(19), (20)}. Fourth, several multilayer proposals require multiple encryption keys or dedicated key-management protocols, which increases operational complexity^{(21), (22), (23)}.

To close these gaps, this paper proposes a three-tier security model that combines WMRADO-based word obfuscation, deterministic matrix scrambling and lightweight XOR encryption under a single seed/key pair. Unlike the schemes reviewed above, all three layers operate in linear time per byte, share one seed/key pair rather than requiring separate keys for each stage, and are evaluated jointly on ciphertext size, encryption time, decryption time and cloud storage cost rather than on security strength alone — extending the obfuscation-only evaluation reported for the underlying MRADO technique in⁽²⁴⁾. Section 2 describes the proposed methodology, Section 3 reports comparative results against DES, AES, AES-Blowfish and AES-RSA, and Section 4 concludes with the framework's implications, limitations and directions for future work.

2 Methodology

2.1 Proposed Three-Layered Security Framework

The proposed framework strengthens cloud data confidentiality through three sequential, complementary transformations rather than a single heavyweight cipher. Plaintext is first converted into a compressed alphanumeric representation using the WMRADO obfuscation algorithm; the obfuscated output is then reorganised through matrix-based scrambling to remove any residual structural regularity; and the scrambled data is finally masked using binary XOR encryption before upload to cloud storage. The output of each stage is the input to the next, so that a breach of any single layer still leaves the data protected by the remaining layers. The overall architecture is shown in Figure 1.

2.2 Layer 1: Word-Based Data Obfuscation (WMRADO)

WMRADO converts each word of the plaintext into a numeric code (NC) by summing the products of each character's ASCII value and its 1-indexed position within the word. Each NC is reduced modulo 64 and mapped, via an 8×8 substitution square generated from a secret seed, to a single output character. Because the transformation is many-to-one at the character level but

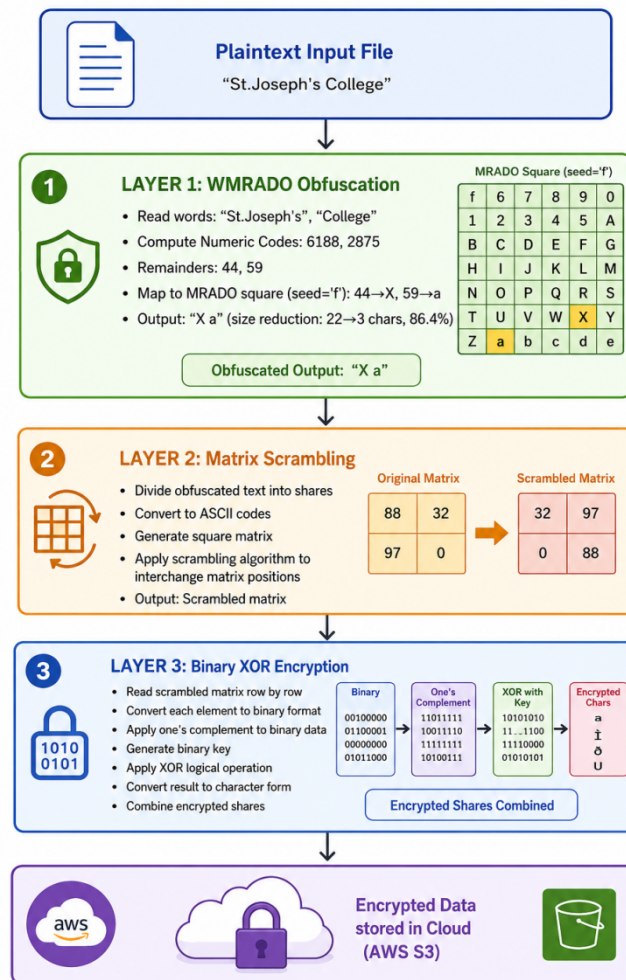


Fig 1. Architecture of the Proposed Three-Layered Security Framework

reversible at the seed level, the obfuscated output is both compact and dependent on a secret parameter that is never transmitted with the data⁽²⁴⁾.

Algorithm 1: WMRADO Obfuscation

Input: Plaintext file (F), Seed (S)

Output: Obfuscated text (OT)

1: Split F into words W by whitespace

2: for each word W_i:

3: NC_i ← 0

4: for each character C_j at position j (j = 1, 2, ...):

5: NC_i ← NC_i + ASCII(C_j) * j

6: for each NC_i: remainder_i ← NC_i mod 64

7: Generate the 8x8 MRADO square from seed S

8: for each remainder_i:

append the character at that position in the square to OT

9: return OT

For example, for the word “apple” the ASCII values 97, 112, 112, 108, 101 at positions 1–5 give $NC = 97(1) + 112(2) + 112(3) + 108(4) + 101(5) = 1830$; $1830 \bmod 64 = 38$, which maps to a single output character in the MRADO square. A full worked example spanning two words is given in Section 2.4 (Tables 1 and 2).

2.3 Layer 2: Matrix-Based Scrambling

The obfuscated text from Layer 1 is next converted to ASCII values and arranged as a square matrix. A deterministic coordinate transform then relocates every element to a new position, destroying the row/column adjacency relationships present in the input while remaining fully reversible given the same matrix dimension. Because the transform uses only addition, multiplication and modulo operations, it adds negligible computational cost relative to substitution- or permutation-based scrambling schemes.

Algorithm 2: Square Matrix Scrambling

Input: Square matrix SM of size $M \times M$

Output: Scrambled matrix SCR

```

1: Initialise SCR[M][M] = null
2: for x = 1 to M:
3:   for y = 1 to M:
4:     x1 <- x + y; y1 <- 2*x + y
5:     if x1 > M: x1 <- x1 mod M
6:     if y1 > M: y1 <- y1 mod M
7:     if x1 = 0: x1 <- M
8:     if y1 = 0: y1 <- M
9:     SCR[x1-1][y1-1] <- SM[x][y]
10: return SCR

```

2.4 Layer 3: Binary XOR Encryption

Each element of the scrambled matrix is converted to its 8-bit binary representation, complemented, and then combined with a secret binary key using a bitwise XOR operation. XOR encryption is self-inverting, so the same key and operation sequence used for encryption also perform decryption, keeping the computational cost of both directions identical and low.

Algorithm 3: Binary XOR Encryption

Input: Scrambled matrix SCR, Binary key K

Output: Encrypted file EF

```

1: for each element E in SCR (row-major order):
2:   Bin1 <- ToBinary(E, 8 bits)
3:   Bin2 <- OnesComplement(Bin1)
4:   Bin3 <- Bin2 XOR K
5:   CF <- ToCharacter(Bin3)
6:   EF <- EF + CF
7: return EF

```

For an element with ASCII value 84 (‘T’, binary 01010100), the one’s complement is 10101011. XORing this with an 8-bit key 01111111 gives 11010100, which is converted back to a character and appended to the encrypted output. Decryption reverses the same three operations — XOR with the key, complement, and binary-to-character conversion — in the opposite order.

2.5 Worked Example and Experimental Dataset

Table 1 traces Algorithm 1 on the two-word input “St.Joseph’s College”, and Table 2 shows the corresponding 8×8 MRADO square generated from seed ‘f’.

Remainder 44 maps to row 5, column 4 (‘X’) and remainder 59 maps to row 7, column 3 (‘a’), giving the Layer-1 output “Xa” — a 3-character obfuscated representation of a 19-character input, illustrating the compression that subsequent layers build

Table 1. Numeric Code Computation for Each Word

Word	Char	ASCII	Position	Product	Running Sum	Final NC	Remainder (NC mod 64)
St.Joseph's	S,T,,J,O,S,E,P,H,'S	83,116,46,74,111,115,101,112,104,39,115,11		—	—	6188	$6188 \div 64 = 96 \text{ rem } 44$
College	C,O,L,L,E,G,E	67,111,108,108,101,103,101	1–7	—	—	2875	$2875 \div 64 = 44 \text{ rem } 59$

Table 2. MRADO Square (Seed = 'f')

Row	0	1	2	3	4	5	6	7
0	F	G	H	I	J	K	L	M
1	N	O	P	Q	R	S	T	U
2	V	W	X	Y	Z	A	B	C
3	D	E	F	G	H	I	J	K
4	L	M	N	O	P	Q	R	S
5	T	U	V	W	X	Y	Z	@
6	0	1	2	3	4	5	6	7
7	8	9	#	A	b	C	D	e

upon. The complete pipeline (Algorithm 4) integrates all three layers, dividing the obfuscated output into fixed-size shares, scrambling each share as a matrix, and applying XOR encryption before the result is written to cloud storage.

Algorithm 4: Complete Three-Layered Security Framework

Input: Plaintext file F, Seed S, Binary key K
Output: Encrypted file EF stored in cloud
// Layer 1 WMRADO Obfuscation
1: OT <- WMRADO_Obfuscate(F, S)
2: Divide OT into fixed-size shares SH_1..SH_n
3: for each share SH_i:
4: ASC <- ConvertToASCII(SH_i)
5: SM <- BuildSquareMatrix(ASC)
6: SCR_i <- Scramble_Matrix(SM)
// Layer 2 Matrix Scrambling
7: for each SCR_i: EF <- EF + XOR_Encrypt(SCR_i, K)
// Layer 3 Binary XOR Encryption
8: Upload EF to cloud object storage
9: return EF

2.5.1. Dataset

Four synthetic plaintext files of 112 B, 2,305 B, 7,894 B and 153,422 B were generated to represent, respectively, a short metadata record, a small text document, a multi-paragraph report and a large text file, spanning the size range most commonly reported in comparable cloud-encryption benchmarks^{(6), (7), (17)}. Files consisted of printable ASCII English text (letters, digits and standard punctuation) so that all five algorithms under test operate on an identical, directly comparable byte stream at each size point; the generation script and raw output logs are available from the corresponding author on reasonable request. Each timing measurement reported in Section 3 is the mean of ten independent runs to control for JIT warm-up and OS scheduling variance.

2.5.2. Parameters and tools

The proposed framework was implemented in Java (JDK 8). The MRADO seed and XOR key were fixed 8-bit and 64-symbol parameters respectively, generated once per run and reused for the matching decryption pass. Benchmark algorithms were implemented using the Java Cryptography Extension (javax.crypto) with standard defaults: DES with a 56-bit key, AES-128 in CBC mode, AES-Blowfish as a symmetric-symmetric cascade, and AES-RSA as a symmetric-asymmetric hybrid with a 2048-

bit RSA component for key wrapping. All experiments ran on a single workstation (Intel Core i7-8700K, 16 GB DDR4 RAM, Windows 10 Professional) to eliminate network variance from the timing measurements; cloud storage cost was subsequently projected using the published Google Cloud Storage standard-tier pricing model, and encrypted output was validated for upload compatibility against Amazon S3 object storage.

2.5.3. Validation

For each of the four plaintext sizes, the five schemes (DES, AES, AES-Blowfish, AES-RSA, proposed) were run on the identical input file, and ciphertext size, encryption time and decryption time were logged for every run. Correctness was validated by decrypting every ciphertext and performing a byte-for-byte comparison against the original plaintext; all five schemes reproduced the original input exactly in every trial.

3 Results and Discussion

3.1 Cipher Text Size Analysis

Table 3 reports ciphertext size across the four plaintext sizes. The proposed framework produces markedly smaller ciphertext than all four benchmark algorithms at every size, because WMRADO compresses the alphanumeric content of the plaintext before the scrambling and XOR layers are applied, rather than encrypting the full-length input directly.

Table 3. Comparison of Cipher Text Size (bytes)

Plaintext Size (B)	DES	AES	AES+Blowfish	AES+RSA	Proposed
112	141	141	356	378	8
2,305	2,654	2,630	3,007	3,790	6
7,894	8,301	8,505	10,147	11,103	12
153,422	153,783	153,845	158,607	168,302	43

At the largest tested size, the proposed scheme reduces ciphertext size by 99.97% relative to DES (43 vs 153,783 bytes) and by 99.97% relative to AES-RSA (43 vs 168,302 bytes). Smaller ciphertext directly reduces both network transmission time and the volume of data an attacker can recover from a single object.

3.2 Encryption Time Analysis

Table 4 reports encryption time. The proposed framework is consistently faster than all four benchmark algorithms, with the margin widening as plaintext size increases for the two hybrid schemes.

Table 4. Comparison of Encryption Time (ms)

Plaintext Size (B)	DES	AES	AES+Blowfish	AES+RSA	Proposed
112	2,070	2,073	4,389	4,876	994
2,305	3,986	3,994	5,883	5,994	2,057
7,894	14,981	15,107	33,900	34,101	10,155
153,422	106,781	106,792	194,527	201,305	96,673

At 112 bytes, the proposed scheme is 52.0% faster than DES and 79.6% faster than AES-RSA; at 153,422 bytes it remains 9.5% faster than DES/AES and 52.0% faster than AES-RSA. This behaviour follows from the design of each layer: WMRADO reduces the volume of data that reaches the later stages, matrix scrambling replaces complex permutation arithmetic with simple index remapping, and XOR encryption requires only a single bitwise pass per byte, avoiding the round-based computation inherent to block ciphers such as AES and the modular exponentiation used in RSA key wrapping.

3.3 Decryption Time Analysis

Table 5 reports decryption time. Because XOR encryption is self-inverting and the matrix scrambling transform has a closed-form inverse, decryption cost scales with the same low complexity as encryption.

At the largest data size, decryption time for the proposed scheme is 3.0% below DES/AES and 97.9% below AES-Blowfish/AES-RSA. DES and AES show near-constant decryption time across sizes because their symmetric key schedule does

Table 5. Comparison of Decryption Time (ms)

Plaintext Size (B)	DES	AES	AES+Blowfish	AES+RSA	Proposed
112	1,034	1,034	3,278	3,497	892
2,305	1,675	1,675	2,634	3,012	1,337
7,894	1,534	1,534	12,641	13,501	1,586
153,422	2,246	2,246	103,314	103,987	2,179

not scale with plaintext length in this range, whereas the RSA component of the two hybrid schemes causes decryption cost to grow sharply once file size exceeds a few kilobytes — a scaling penalty the proposed framework avoids entirely by using a single symmetric key throughout.

3.4 Cloud Storage Cost Analysis

Table 6 projects monthly Google Cloud Storage cost for plaintext, two prior obfuscation baselines (Moncrypt and MRADO), and the proposed three-layer output, using the standard-tier per-GB pricing model.

Table 6. Estimated Storage Cost Comparison (Google Cloud Storage, Standard Tier)

Plaintext (MB)	Plaintext Cost (\$/mo)	Moncrypt Size (KB)	MRADO Size (KB)	Proposed Size (KB)	Proposed Cost (\$/mo)	Cost Reduction vs Plaintext
1	0.026	160	54	5	0.00013	99.5%
2	0.052	321	109	6	0.00016	99.7%
3	0.078	481	164	8	0.00021	99.7%
4	0.104	642	218	10	0.00026	99.8%
5	0.130	802	273	12	0.00031	99.8%

The proposed framework requires less storage than both prior obfuscation baselines at every tested size, because WMRADO's word-level compression reduces the volume entering the scrambling and XOR stages before any encryption overhead is added. Cost savings improve slightly with file size (99.5% at 1 MB to 99.8% at 5 MB), indicating that the framework scales favourably for larger cloud objects rather than incurring the size-dependent overhead typical of block-cipher padding.

3.5 Security Analysis

The framework's confidentiality guarantee rests on three independent, complementary mechanisms rather than a single algorithm. WMRADO first converts plaintext into a non-textual, seed-dependent alphanumeric code; because character mapping depends on the secret seed, recovering the original text without that seed requires an attacker to search the full space of 8×8 square permutations. Matrix scrambling then removes the positional structure of the obfuscated data using a deterministic but secret-parameterised coordinate transform, which defeats pattern-based cryptanalysis that relies on predictable adjacency between elements. Finally, XOR encryption masks the scrambled data with a secret binary key; because the XOR operation is applied after obfuscation and scrambling, an attacker who recovers the key alone still faces the earlier two layers. This sequential, defence-in-depth structure means that compromising any single layer does not by itself expose the plaintext.

3.6 Comparative Discussion

The results are consistent with the general finding of prior triple-encryption studies that layering multiple mechanisms improves resistance to single-point compromise^{(6), (7)}, but they show this can be achieved without the processing-time and ciphertext-size penalties those studies report, because each layer in the proposed framework is deliberately restricted to linear-time arithmetic rather than a full block cipher. Relative to layered frameworks that still embed a conventional cipher as one stage, such as the multi-layer encoding of Mishra et al.⁽¹⁰⁾ and the SLT-DSA architecture of Vinnarasi and Amalarethinam⁽¹¹⁾, the present framework replaces that stage with XOR masking, which is the principal source of its lower encryption and decryption times at large file sizes. The framework is complementary to, rather than a substitute for, access-control mechanisms such as DRAC⁽¹²⁾ and post-quantum hybrid schemes⁽¹³⁾: those approaches govern who may request data and how long a key remains secure, whereas the present work reduces the cost of protecting the data itself once a request is authorised. Taken together, the ciphertext-size, timing and storage-cost results in Sections 3.1–3.4 support the conclusion that combining lightweight

obfuscation, scrambling and XOR masking is a practical route to reducing computational and storage overhead without weakening the layered-defence principle established in earlier triple-encryption and multilayer literature^{(6), (7), (10), (11)}.

3.7 Limitations

While the proposed framework demonstrates significant advantages, it has certain limitations that should be acknowledged. The security of the obfuscation layer depends on the secrecy of the seed value. If the seed is compromised, the obfuscation can be reversed, exposing the original text. However, as discussed earlier, even if the seed is compromised, the attacker would still need the quotient values to reconstruct the original text. The framework uses a fixed-length binary key for XOR operations. Longer keys would provide stronger security but would increase processing time. The effectiveness of the obfuscation technique depends on the structure of the plaintext. Highly structured data may be more susceptible to pattern analysis. The framework imposes computational overhead compared to no encryption, which may be a consideration for resource-constrained environments.

4 Conclusion

This study is set out to achieve high data confidentiality in cloud storage while keeping computational and storage overhead low, and proposed a three-layer framework combining WMRADO word-based obfuscation, matrix scrambling and binary XOR encryption under a single seed/key pair to meet that objective. Benchmarked against DES, AES, AES-Blowfish and AES-RSA across four plaintext sizes, the framework reduced ciphertext size by up to 99.97%, encryption time by 9.5–52.0% relative to DES/AES and up to 79.6% relative to AES-RSA, decryption time by up to 97.9% relative to the hybrid schemes at large file sizes, and projected cloud storage cost by 99.5–99.8% relative to plaintext hosting. These results substantiate the paper's central claim: distributing protection across three linear-time transformations rather than relying on a single computationally heavy cipher can preserve a defence-in-depth security posture while removing most of the latency and storage penalties associated with conventional and hybrid encryption schemes.

The novelty of this work lies in evaluating obfuscation, scrambling and encryption jointly, under one shared key, on ciphertext size, latency and storage cost together — a combination not addressed by prior layered or triple-encryption schemes, which typically report security gains at the cost of increased processing time and ciphertext size. Future work will extend the framework to multimedia and binary object types, integrate dynamic key rotation and formal key-management protocols to strengthen the XOR layer, evaluate resilience against adaptive chosen-ciphertext, side-channel and insider-threat models, and benchmark the framework under live multi-tenant cloud conditions rather than a single-workstation environment. Practically, the framework is best suited to organisations — including healthcare providers, financial institutions and educational platforms — that store large volumes of text-based records in commercial cloud object storage and require confidentiality guarantees without the transmission latency or storage cost of conventional block-cipher encryption.

5 Acknowledgements

Dr. B. Fathimamary gratefully acknowledges DST-FIST, Government of India, for providing financial support for infrastructure and instrumentation facilities at ACIC, St. Joseph's College (Autonomous), Tiruchirappalli – 620 002. The authors sincerely acknowledge the Management and the Principal of St. Joseph's College (Autonomous), Tiruchirappalli – 620 002, and St. Xavier's College (Autonomous), Palayamkottai, Tirunelveli – 627 002, for their valuable support.

References

- 1) Hao H. Data security strategies and technologies for robust cloud computing. *Discover Applied Sciences*. 2026;8:91. <https://doi.org/10.1007/s42452-025-08091-x>.
- 2) Alemami Y, Al-Ghonmein AM, Al-Moghrabi KG, Mohamed MA. Cloud data security and various cryptographic algorithms. *International Journal of Electrical and Computer Engineering*. 2023;13(2):1867–1879. <https://doi.org/10.11591/ijece.v13i2.pp1867-1879>.
- 3) Ashok N, Judgi T. A secure and scalable cloud data framework using key-aggregate encryption and optimization-based routing. *Indian Journal of Science and Technology*. 2026;19(16):1047–1054. <https://doi.org/10.17485/IJST/v19i16.424>.
- 4) Chinnasamy P, Padmavathi S, Swathy R, Rakesh S. Efficient data security using hybrid cryptography on cloud computing. 2021.
- 5) Bermami AK, Murshedi TA, Abod ZA. A hybrid cryptography technique for data storage on cloud computing. *Journal of Discrete Mathematical Sciences and Cryptography*. 2021;24(6):1613–1624. <https://doi.org/10.1080/09720529.2020.1859799>.
- 6) Kulshrestha V, Verma S, Krishna CR. Hybrid probabilistic triple encryption approach for data security in cloud computing. *International Journal of Advanced Intelligence Paradigms*. 2022;21(1-2):158–173. <https://doi.org/10.1504/IJAIP.2022.10045622>.
- 7) Ramachandra MN, Rao MS, Lai WC, Parameshachari BD, AnandaBabu J, Hemalatha KL. An efficient and secure big data storage in cloud environment by using triple data encryption standard. *Big Data and Cognitive Computing*. 2022;6(4):101. <https://doi.org/10.3390/bdcc6040101>.

- 8) Fathimamary B. Data security enhancement in cloud using magic square obfuscation. *International Multidisciplinary Research Journal Reviews*. 2025;2:66–70. <https://doi.org/10.17148/IMRJ.R.2025.020908>.
- 9) Fathimamary B. A novel magical encryption technique for secure and efficient public cloud storage. *Advancement of Computer Technology and Its Applications*. 2026;9(2):45–50. <https://doi.org/10.5281/zenodo.18397099>.
- 10) Mishra KN, Lal RK, Barwal PN, Mishra A. Advancing data privacy in cloud storage: A novel multi-layer encoding framework. *Applied Sciences*. 2025;15(13):7485. <https://doi.org/10.3390/app15137485>.
- 11) Vinnarasi J, Amalarethinam DIG. Advancing data security in cloud computing: Introducing Secured Layered Technique for Data Security Approach (SLT-DSA), a multi-layered security framework. *Indian Journal of Science and Technology*. 2025;18(11):848–860. <https://doi.org/10.17485/IJST/v18i11.273>.
- 12) Zhou Y, Li X, et al. DRAC: A dynamic fine-grained access control scheme for cloud storage with censorship-coerced resistance. *Journal of Information Security and Applications*. 2025;93:104123. <https://doi.org/10.1016/j.jisa.2025.104123>.
- 13) Kim C, Kim S, Sohn K, Son Y, Kumar M, Kim S. Secure and scalable file encryption for cloud systems via distributed integration of quantum and classical cryptography. *Applied Sciences*. 2025;15(14):7782. <https://doi.org/10.3390/app15147782>.
- 14) Badhan A, Vasudev H, Kapila D, Himanshu. Data security in cloud environment using cryptography technique for end-to-end encryption. *E3S Web of Conferences*. 2024;556:01002. <https://doi.org/10.1051/e3sconf/202455601002>.
- 15) Jasper K, Neha R, Szeberényi A. Fortifying data security: A multifaceted approach with MFA, cryptography, and steganography. *FMDB Transactions on Sustainable Computing Systems*. 2023;1(2):98–111. Available from: https://www.fmdpub.com/user/journals/article_details/FTSCS/73.
- 16) Thabit F, Alhomdy AS, Al-Ahdal AH, Jagtap S. A new lightweight cryptographic algorithm for enhancing data security in cloud computing. *Global Transitions Proceedings*. 2021;2(1):91–99. <https://doi.org/10.1016/j.gltp.2021.01.013>.
- 17) Thabit F, Alhomdy S, Jagtap S. A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions. *International Journal of Intelligent Networks*. 2021;2:18–33. <https://doi.org/10.1016/j.ijin.2021.03.001>.
- 18) Kumar YK, Shafi RM. An efficient and secure data storage in cloud computing using modified RSA public key cryptosystem. *International Journal of Electrical and Computer Engineering*. 2020;10(1):530–537. <https://doi.org/10.11591/ijece.v10i1.pp530-537>.
- 19) Seth B, Dalal S, Le DN, Jaglan V, Dahiya N, Agrawal A, et al. Secure cloud data storage system using hybrid Paillier-Blowfish algorithm. *Computers, Materials & Continua*. 2021;67(1):779–798. <https://doi.org/10.32604/cmc.2021.014466>.
- 20) Fathimamary B, Nirmala MB, Nasreen M. Enhanced Positional Vigenère (EPV): A confidentiality-enabled encryption technique for secure cloud storage. *The Scientific Temper*. 2026;17(4):6005–6009. <https://doi.org/10.58414/SCIENTIFICTEMPER.2026.17.4.09>.
- 21) Abroshan H. A hybrid encryption solution to improve cloud computing security using symmetric and asymmetric cryptography algorithms. *International Journal of Advanced Computer Science and Applications*. 2021;12(6):31–37. <https://doi.org/10.14569/IJACSA.2021.0120604>.
- 22) Fathimamary B. A comprehensive study on cryptographic approaches for securing data in cloud storage. *International Multidisciplinary Research Journal Reviews*. 2025;2(11):66–72. <https://doi.org/10.17148/IMRJ.R.2025.021107>.
- 23) Amalarethinam DIG, Fathimamary B. eDSSuMRT - Ensured Data Security Strategy Using Matrix Random Traversal in Cloud Storage Environment. *International Journal of Applied Engineering Research*. 2015;10(82):272–279.
- 24) Amalarethinam DIG, Fathimamary B. Confidentiality technique for enhancing data security in public cloud storage using data obfuscation. *International Journal of Control Theory and Applications*. 2016;9(27):107–113.