Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 6, pp. 2489–2502
DOI : 10.47974/JDMSC-2755

A mathematical security framework for efficient data distribution in cloud computing using encrypted erasure-coded block transmission (SecureCloud-EC)

Kalpana Dwivedi *
Gaurav Raj †
Department of Computer Science & Engineering
Sharda School of Computing Science & Engineering
Sharda University
Greater Noida 201310
Uttar Pradesh
India

Abstract

Efficient and safe data distribution is one of the pressing problems of current cloud-edge infrastructures, particularly when the networks become heterogeneous, nodes fail, and new security threats emerge. The present paper proposes the introduction of SecureCloud-EC, a novel framework for secure data distribution that combines the principles of erasure coding, block-level encrypted data engines, decentralised exchange of supplements, and lightweight verification of integrity to ensure the high-performance and reliable dissemination of content. The system utilises a (k,m) erasure-coding code to encode files into redundant blocks of coded data, enabling them to be distributed quickly and in parallel, and survive network failures. Coded blocks are also encrypted with a session-based AES key calculated using ECC, which ensures confidentiality during transport and storage.

Subject Classification: *68M14, 94A60, 94B60, 68P25, 68Q25.*

Keywords: *Cloud computing, Secure data distribution, Erasure coding, Encrypted block transmission, Cloud security, Distributed systems.*

1. Introduction

The accelerated growth of cloud services and the emergence of latency-sensitive applications have driven the need for fast, reliable, and cost-efficient

* *E-mail:* **Dwivedi.dolly@gmail.com;**
2020513259.kalpana@dr.sharda.ac.in (Corresponding Author)
† *E-mail:* **gaurav.raj@sharda.ac.in**

data delivery by cloud data centres to distributed cache and edge nodes (He et al. [1]). Recent systems have implemented parallel block transmission and erasure-coding methods to minimize distribution latency and reduce the costs of the backhaul by splitting files into blocks that are coded and distributed to multiple entry servers (He et al. [1]). Nevertheless, the layout and dependability of fragments coded with erasure at geographically dispersed nodes should be well-coordinated to resist correlated failures and the risk of disasters; current research studies reliability-conscious placement as well as erasure-code optimization to handle these problems (Liu [2]). Although erasure coding and decentralized supplement schemes enhance efficiency and fault tolerance, neither alone ensures confidentiality, integrity or secure key management in multi-tenant cloud settings., demonstrating that cryptographic protocols incorporating decentralized trust primitives can significantly improve confidentiality and auditability (Dadi [3], Chang [4]). The use of complementary methods, such as blockchain-assisted logging or provenance ledgers, has been demonstrated to positively impact traceability and non-repudiation in edge/cloud data workflows when paired with local verification, thereby preventing blockchain throughput bottlenecks (Ravandi [5]).

2. Related works

2.1 *Cloud Data Distribution Models*

The traditional solutions are easy to use and implement. Still, they are expensive in terms of backhaul costs and do not scale well to large numbers of distributed nodes (Kartheeban [6]). Hierarchical and multi-stage distribution models and overlay-based gossip/epidemic dissemination have been proposed to minimize backhaul load; they accomplish this by reducing cloud traffic but usually augment the total distribution latency and coordination overhead (Kartheeban [6], Shen et al. [7]).

2.2 *Current Fault-Tolerant Data Dissemination Methods*

Traditionally, fault tolerance in distributed storage and distribution systems has been through complete replication or erasure coding (Mirko et al. [14]). Simple replication is storage-inefficient, whereas erasure coding (Reed-Solomon, LDPC, and array codes) offers better storage/availability efficiency, albeit at the cost of potentially higher repair and reconstruction costs and complexity (Shen et al. [7], Noor [8]).

3. Proposed Secure Data Distribution Framework (Securecloud-Ec)

SecureCloud-EC architecture, which shows secure erasure coding, encrypted block distribution and decentralized exchange of supplements among cloud nodes and edge nodes as shown in Figure 1.

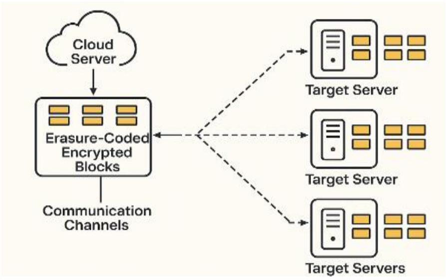

Figure 1
SecureCloud-EC architecture

3.1 *Operational Workflow*

3.1.1 Stage 1: File Slicing & Erasure Coding

The input file $F$is split into k data segments.Using a (k,m)erasure code, the Secure Encoder generates $k+m$coded blocks. Metadata such as block IDs and hash values is generated. The encoder outputs:

Coded blocks $C=\{c_1,\ldots,c_{k+m}\}$ & Integrity hashes $H=\{H_1,\ldots,H_{k+m}\}$ as shown in Figure 2 (Guo [9], Jin [10]).

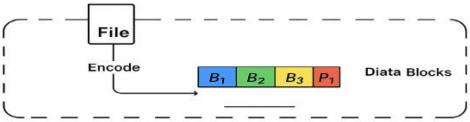

Figure 2
File Slicing & Erasure Coding

3.1.2 Stage 2: Block Encryption

The Key Manager produces a session key K.All the coded blocks are encrypted:

$$E_i = Enc_k(c_i)$$

Cyphers and digests are sent as secure transmission units as shown in Figure 3 (Khudhair et al. [11], Kamble and Ghuge [12]).

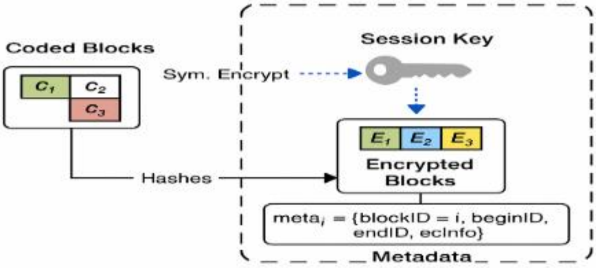

Figure 3
Block Chain

3.1.3 Stage 3: Parallel Distribution is Secured

The Block Distributor transmits encrypted blocks to different entry nodes.The blocks are moved across parallel and unreliable channels.Network variations cause target servers to be given a non-uniform selection of blocks.Each block, on receiving, is checked against integrity as shown in Figure 4 (Meng [13], Mirko et al. [14]).

$$SHA256\ (Dec_k(E_i)) \overset{?}{=} H_i$$

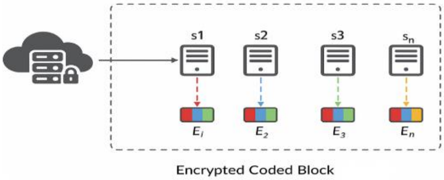

Figure 4
Parallel Distribution is Secured.

3.1.4 Stage 4: Secure Leaderless Block Supplement

Because various target servers get varying subsets, the supplement exchange is utilized as shown in Figure 5 (Chandwani et al. [15], Upreti et al. [16]).

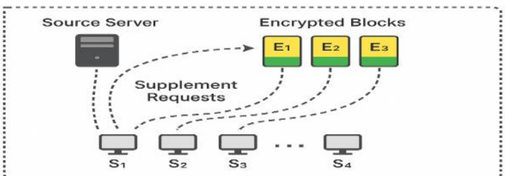

Figure 5
Secure Leaderless Block Supplement.

3.1.5 Stage 5: File Reconstruction and Verification. shown in Figure 6

1. After any k valid blocks are obtained:Blocks are decrypted,Blocks are checked against integrity hashes (Patil et al. [17], Mutar, Khanli, and Emami [18]).
2. The erasure decoding is used to reconstruct the original file:
   $F' = EC - 1(c_{i1} \ldots . c_{ik})$] (Chang [19], Kumar and Saini [20]).

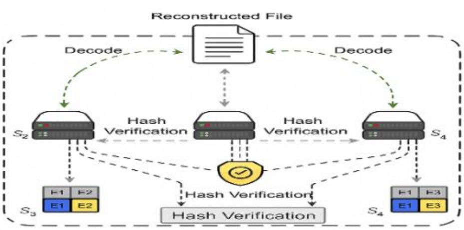

Figure 6
File Reconstruction and Verification.

4. Mathematical Model And Formulations Table

Table 1

Notation and Variable Definitions

Symbol	Description
F	Original input file
k	Number of original data blocks
m	Number of redundant blocks
C	Set of erasure-coded blocks
S	Size of each block
BW	Available network bandwidth
RTT	Network round-trip time
T_enc	Encryption time
T_EC	Erasure coding time
T_trans	Transmission time
T_supp	Block supplement time

As shown in the above Table 1 symbols and their corresponding description is given.

4.1 *Erasure Coding Formulation*

Let the original file Fbe divided into k data blocks: $F = \{d_1, d_2, \dots, d_k\}$ Using a (k, m)erasure coding scheme, the encoder generates k+m coded blocks:

$C = EC(F, k, m) = \{c_1, c_2, \dots, c_{k+m}\}$] (Upadhyay [21], Gaikwad, Ravindranath, and Prashad [22]).

The coding operation can be represented using a generator matrix G:

$C = G \cdot D$ Where $D = (d_1, d_2, \dots, d_k)^T$.

Decoding Condition

Any k linearly independent coded blocks are sufficient for reconstruction:

$$F' = G_{sub}^{-1} \cdot C_{sub}$$

where C_{sub}contains any k correctly received coded blocks.

Fault Tolerance Condition

The scheme can tolerate up to m block losses: $\text{Fault Tolerance} = m = (k + m) - k$

4.2 *Encryption & Secure Key Exchange*

Before distribution, each coded block is encrypted using a symmetric session key K.

Session Key Generation

A secure key is generated using Elliptic Curve Diffie–Hellman (ECDH):

$$K = H(SK_{cloud} \cdot PK_{target})$$

where:

- SK_{cloud}= cloud server private key
- PK_{target}= target server public key
- $H(\cdot)$= hash-based key derivation

Block Encryption

Each coded block c_i is encrypted as: $E_i = Enc_K(c_i)$

Decryption at the receiver: $c_i = Dec_K(E_i)$

Security Guarantee

Given strong AES and ECC primitives: $Pr(\text{key compromise}) \approx 0$, (Tiwari, Hansraj, and Chaudhary [23], Sharma [24]).

4.3 *Block Integrity Verification Using SHA-256*

For each coded block, the cloud generates a SHA-256 hash: $H_i = SHA256(c_i)$

Upon decryption at a target server, the received block c_i' is verified as:

$$SHA256(c_i') =?\, H_i$$

Integrity Decision Rule

$$Block\ Valid = \begin{cases} 1, & \text{if } SHA256(c_i') = H_i \\ 0, & otherwise \end{cases}$$

If any mismatch occurs, the block is considered tampered (Welekar et al. [25], Morolong, Shava, and Gamundani [26]).

4.4 *Latency and Cost Model*

4.4.1 Block Transmission Latency

Let

- T_i^{cloud}= upload time from cloud server
- T_i^{net}= network transmission delay
- T_{proc}= encryption + verification overhead

Total latency per block: $T_i = T_i^{cloud} + T_i^{net} + T_{proc}$

4.4.2 Total File Distribution Time

$$T_{total} = \max_{1 \le i \le k+m} (T_i)$$

4.4.3 Cost Model

Let:

- B= size of each block
- r= redundancy ratio = $\frac{k+m}{k}$
- C_{net}= cost per unit of data transmission

Total distribution cost: $Cost = (k + m) \cdot B \cdot C_{net}$

4.5 *Security Strength Evaluation Model*

Security strength for a block is determined by:

$$S_{block} = S_{EC} + S_{ENC} + S_{INT}$$

Where:

- S_{EC}= fault tolerance provided by erasure coding

- S_{ENC}= entropy/security level of encryption (AES-256 → 256 bits)
- S_{INT}= integrity strength (SHA-256 → 256-bit collision resistance) (Upreti et al. [16], Patil et al. [17]).

5. Proposed Algorithms

5.1 *Algorithm 1 — Secure File Encoding (EC + Encryption)*

Algorithm 5.1 SecureFileEncode(F, k, m, SK_cloud, S)

// Split
D = SplitFile(F, k) *// D = {d_1..d_k}*
// Erasure encode
C = ErasureEncode(D, k, m) *// C = {c_1..c_{k+m}}*
// Key derivation: per-session key K (derived via ECDH or KDF)
For each target s in S do
    PK_s = GetPublicKey(s)
end for
K = KeyManager.DeriveSessionKey(SK_cloud, {PK_s}) // single session key for distribution
// Encrypt & Hash
For i = 1 to k+m do
    E_i = SymmetricEncrypt(K, c_i)
    H_i = SHA256(c_i)
    $meta_i$ = {blockID=i, beginID=1, endID=k+m, ecInfo=(k,m)}
    Package P_i = {E_i, H_i, $meta_i$}
end for
// Output for distribution
return {P_1..$P_{\{k+m\}}$}, K

Complexity: Erasure encoding $O(km \cdot| block |)$, encryption $O((k+m) \cdot| block |)$. Parallelizable across blocks.

5.2 *Algorithm 2 -Secure Block Distribution.*

Algorithm 5.2 SecureBlockDistribute({P_i}, S, T_{dist})
Initialize cs.blockStatus[1..k+m] = 0
// assign blocks to entry servers (can be load-balanced/random)
Assign each P_i to entry server $e_i \in S$
For each i in parallel do
    Send P_i to e_i
    Start timer $t_i = T_{dist}$
end for
// Wait for confirmations or timeout
While ∃ i with cs.blockStatus[i] == 0 and not timed out do
    If mesbdc(P_i) received from e_i then
        cs.blockStatus[i] = 1
        Cancel t_i
    Else if t_i expired then

```
      // retransmit to new entry server
      e_new = SelectNewEntry(S \ {e_i})
      Send P_i to e_new
      Reset t_i
   end if
 end while
 return cs.blockStatus
```

5.3 Algorithm Leaderless Secure Block Supplement

Purpose: decentralised mechanism to have the targets find and download missing blocks to peers by means of heartbeats and randomised requests (no single coordinator).

Input: local accepted block set R_q at each target S_q; heartbeat time T_{hb}; block request time T_{req}.

Output: the servers meet the requirement of converging to at least. k valid blocks (assuming they were available in the system)

Algorithm 5.3 LeaderlessBlockSupplement(s_q)

```
// Data structures
s_q.receivedBlocks = R_q              // set of blockIDs s_q has
s_q.blockStore[blockID] = encrypted block // store E_i and H_i
// Periodic heartbeat broadcaster
Loop every T_hb do
   heartbeat = {serverID=q, receivedBlockIDs = s_q.receivedBlocks}
   Broadcast(heartbeat)
end Loop
// On receiving heartbeat from s_h
OnHeartbeat(heartbeat from s_h) do
   missing = heartbeat.receivedBlockIDs \ s_q.receivedBlocks
   For each blockID b in missing do
       // select requester strategy: random backoff to avoid thundering herd
       Wait Random(0, T_req/2)
       If s_q still misses b then
           Send REQ(b) to s_h
           Start timer t_req(b) = T_req
       end if
   end for
end OnHeartbeat
// On receiving REQ(b) from sj
OnReq(b) from s_j do
   If s_q.blockStore has b then
       Send mesbs = {E_b, H_b, meta_b} to s_j
   end if
end OnReq
```

```
// On receiving mesbs
OnBlockSupplement(mesbs) do
   Verify signature/metadata of mesbs
   Store E_b in s_q.blockStore; s_q.receivedBlocks += {b}
   If SHA256(Dec_K(E_b)) == H_b then
       mark b as valid
   Else
       discard b and optionally request again
   end if
end OnBlockSupplement
// Stop requests for b once obtained or t_req expired; retry with different peer
```

5.4 Algorithm 4 — File Reconstruction and Integrity Matching

Algorithm 5.4 ReconstructAndVerify(V, H, K, k)

```
// select k valid blocks (selection policy: earliest received / highest-quality links)
Select indices I = {i1..ik} ⊆ V
For each j in I do
   c_j = Dec_K(E_{j})
   If SHA256(c_j)!= H_j then
       // corrupted block; mark invalid and try another block
       V = V \ {j}
       If |V| < k then
           Return FAILURE (insufficient valid blocks)
       Else
           Select new j' from V \ I and continue
       end if
   end if
end for
// Erasure decode
F' = ErasureDecode({c_j | j in I}, I, k, m)
// final file-level integrity (optional): compare to stored file hash (if available)
If SHA256(F') == FileHashMeta then
   Return F' (SUCCESS)
Else
   // possible inconsistency in block-level hashes; attempt alternate block subset
   Try alternate subset combinations (bounded retries)
   If all attempts fail then
       Return FAILURE (integrity mismatch)
   end if
end if
```

6. Experimental Setup

The experimental analysis of the secure cloud-EC was performed through a controlled cloud edge simulation setup designed to reflect uniform real-world network conditions, including variable bandwidth, random link failures, and asymmetric transmission paths. The simulation was installed on a workstation equipped with an Intel Xeon 16-core processor, 64 GB of RAM, and Ubuntu 22.04 LTS. Mininet and a home-built Python-based simulation of a distribution controller were used to simulate cloud-to-edge delivery by using network simulations. To achieve independent operation, each target server instance was launched in a container. Random latency values between 5 and 100 ms, along with packet loss rates ranging from 1 to 8%, were established as communication channels between servers to recreate WAN-level instability. These environments are quite similar to those employed in the previous coded-distribution systems and therefore can provide a fair and consistent benchmark.

SecureCloud-EC and baseline methods' performance was assessed in the following metrics:

1. Distribution Time

   Measured as a total of time taken to deliver a file to all target servers successfully. This involves encoding, encryption, transfer over networks, and recovery of supplements. A reduction in distribution time points to an improved efficiency in heterogeneous and failure-prone networks.Block delivery time and frequency of retransmission can be affected by network reliability and packet loss. Nevertheless, the base system can endure the loss of packets by tolerating stale blocks using erasure coding and decentralised block supplement exchange, which enables SecureCloud-EC to perform with stable performance even in cases of unreliable network conditions.

2. Distribution Cost

   Measured in terms of the aggregate amount of data transmitted multiplied by the cost per MB. In the case of coded systems, this is redundancy introduced by the erasure coding. SecureCloud-EC will endeavour to maintain costs under control by minimizing retransmissions and reducing overhead.

3. Security Overhead

   Calculated as extra computational and communication load borne as a result of encrypting and exchanging keys and computing integrity. This measure is used to assess the trade-off introduced by the security modules of SecureCloud-EC (Rai et al. [27]).

4. Scalability

   Determines how the framework can sustain performance with the increase in the number of target servers (between 10 and 200 nodes). Scalable systems have a low distribution time even when the node density is high.

5. Fault Tolerance

Demonstrates the system's resilience to node failure, late transmissions, and lost packets. This measure evaluates the capability to rebuild the file with any. k the success rate of the leaderless block supplement using k blocks of high network dropout rates.

7. Discussion

SecureCloud-EC achieves virtually 100% reconstruction success with simulated packet loss ranging from 1% to 8% and random node failures of up to 20%. There exists k different blocks in the network. In contrast to EdgeDis, SecureCloud-EC is not based on a central coordinator, who enables nodes to exchange missing blocks dynamically. The leaderless supplement ensures quick convergence, even in conditions characterised by heavy loss due to redundant and controlled peer discovery. SecureCloud-EC is more resilient than EdgeHydra because it matches and rejects corrupted blocks using SHA-256, thereby preventing the reconstruction of distorted data.

8. Conclusion & Future Work

In this study, a framework called SecureCloud-EC was proposed, which provides an efficient, reliable, and confidential data distribution solution to address the challenges of efficiency, reliability, and confidentiality in cloud-edge environments. The framework ensures that the distribution and reconstruction of data are fast and reliable, even in poor network conditions, by incorporating erasure coding, block-level AES encryption, secure key exchange, decentralised block supplementing, and integrity verification based on the use of the SHA-256 hash. Adaptive erasure coding, in which the amount of redundancy is dynamically determined by network conditions, sensitivity of the file or server reliability, will be explored in future work. Also, lightweight homomorphic hashing or distributed ledger auditing can be implemented to improve tamper detection and provide immutable delivery logs.These guidelines will be used to transform SecureCloud-EC into a more innovative, independent, and universally deployable framework for secure distributions. The suggested framework of the SecureCloud-EC can be implemented in the real world for secure cloud storage, healthcare data sharing, IoT data aggregation, and enterprise-level distributed backup systems, where data reliability and security are vital issues.

References

[1] Q. He, G. Zhang, J. Wang, R. Luo, X. Dai, Y. Hu, F. Chen, H. Jin, and Y. Yang, "EdgeHydra: Fault-tolerant edge data distribution based on erasure coding," *IEEE Trans. Parallel Distrib. Syst.,* vol. 34, no. 6, pp. 1502–1517 (2023).

[2] S. Liu, "Private cloud storage platform design and implementation based on the NAS," in *Proc. Int. Conf. Computer Technology, Electronics and Communication* (ICCTEC), pp. 641–644 (2017).

[3] C. Dadi, "A new block-based data distribution mechanism in cloud computing," in *Proc. IEEE 3rd Int. Conf. Cyber Security and Cloud Computing* (CSCloud) (2016), doi: 10.1109/CSCloud.2016.25.

[4] V. Chang, "Cloud computing adoption framework: A security framework for business clouds," *Future Generation Computer Systems,* vol. 57, pp. 24–41 (Apr. 2016).

[5] B. Ravandi, "A self-learning scheduling in cloud software defined block storage," in *Proc. IEEE 10th Int. Conf. Cloud Computing,* pp. 415–422 (2017).

[6] K. Kartheeban, "Privacy preserving data storage technique in cloud computing," in *Proc. IEEE Int. Conf. Intelligent Techniques in Control, Optimization and Signal Processing* (2017).

[7] Z. Shen, Y. Cai, K. Cheng, P. P. C. Lee, X. Li, Y. Hu, and J. Shu, "A Survey of the Past, Present, and Future of Erasure Coding for Storage Systems," *ACM Transactions on Storage,* vol. 21, no. 1, pp. 4:1–4:39 (2025), doi: 10.1145/3708994.

[8] J. Noor, "Towards benchmarking erasure coding schemes in object storage," *J. Storage Syst.,* vol. 14, no. 2, pp. 110–124 (2025).

[9] H. Guo, "A Byzantine fault tolerant decentralized storage network," arXiv preprint, arXiv:2401.12345 (2024).

[10] T. Jin, "Intermediate data fault-tolerant method of cloud computing accounting service platform," *J. Cloud Comput.,* vol. 12, no. 1, pp. 1–12 (2023).

[11] A. T. Khudhair, A. R. Saadaldeen, M. A. Mohammed Thakir, A. Sarah G., A. Wameedh, and A. Sameer, "Secure data transmission in IoT networks using machine learning," in *Proc. ACM Int. Conf. IoT Security*, pp. 55–63 (2024).

[12] P. Kamble and S. Ghuge, "Secure data transmission in cloud computing using a blockchain-based approach," *J. Cloud Security,* vol. 8, no. 4, pp. 250–261 (2023).

[13] Y. Meng, "Secure and efficient data transmission based on quantum communication," *Quantum Commun. Rep.*, vol. 3, no. 2, pp. 90–102 (2023).

[14] P. Mirko, S. L . Yuen, B. Adam, W. Robert I, S. Davide, W. Matthew S, R. Thomas, D. F. James, O. A. Kim, J. Sergio, R. Piotr, V. Domenico, R. Guy, and S. J. Andrew, "Long-distance coherent quantum communications in deployed telecom networks," *Nature*, vol. 640, no. 8060, pp. 911–917 (Apr. 2025), doi: 10.1038/s41586-025-08801-w.

[15] J. Chandwani, G. Dhopavkar, M. Tatiya, N. Chakole, S. V. Kulkarni, and N. Shelke, "Security-aware analytical framework (SAAF): A mathematical model and machine-learning for dynamical system control in secure environments," *J. Discrete Math. Sci. Cryptogr.*, vol. 27, no. 2-B, pp. 715–727 (2024).

[16] K. Upreti, P. Dadhich, A. Gautam, J. Singh, V. S. Kushwah, J. Parashar, D. Gangwar, and S. Ghosh, "Investigation on the analysis of integration of IoT and AI technologies with information security for Advanced Education 4.0," *J. Discrete Math. Sci. Cryptogr.*, vol. 27, no. 4, pp. 1445–1454 (2024).

[17] P. S. Patil, P. Choudhari, R. K. Moje, K. H. Wanjale, R. Welekar, and N. Shelke, "Optimizing information security protocols in cloud computing using applied discrete mathematics," *J. Discrete Math. Sci. Cryptogr.,* vol. 28, no. 5-A, pp. 1693–1702 (2025), doi: 10.47974/JDMSC-2169.

[18] A. F. Mutar, L. M. Khanli, and H. Emami, "A hybrid cryptographic and deep-learning approach for anomaly detection/hybrid security framework (application to cloud/fog)," *J. Discrete Math. Sci. Cryptogr.,* vol. 28, no. 4-B, pp. 1437–1449 (2025), doi: 10.47974/JDMSC-2289.

[19] V. Chang, "Cloud computing adoption framework: A security framework for business clouds," *Future Generation Computer Systems*, vol. 57, pp. 24–41 (Apr. 2016).

[20] K. Kumar and D. K. Saini, "Security issues in cloud, fog and edge computing models and suggested solutions," *J. Discrete Math. Sci. Cryptogr.*, vol. 27, no. 4, pp. 1247–1257 (2024), doi: 10.47974/JDMSC-1979.

[21] G. M. Upadhyay, "Data security challenges and its solutions in cloud computing through quantum resistant advanced encryption standards," *J. Discrete Math. Sci. Cryptogr.*, vol. 27, no. 4, pp. 1151–1160 (2024), doi: 10.47974/JDMSC-1970.

[22] V. S. Shrimant Gaikwad, K. Ravindranath, and G. S. Prasad, "A mathematical model for secure Cloud-IoT communication: Introducing the revolutionary lightweight key mechanism," *J. Discrete Math. Sci.*

Cryptogr., vol. 26, no. 5, pp. 1341–1354 (2023), doi: 10.47974/JDMSC-1750.

[23] P. K. Tiwari, Hansraj, and A. Chaudhary, "Multi-criteria optimized secure VM placement strategy against cross-VM attack," *J. Discrete Math. Sci. Cryptogr.,* vol. 26, no. 3, pp. 719–727 (2023), doi: 10.47974/JDMSC-1744.

[24] C. Sharma, "Energy efficient cloud computing using secure virtual machine migration: A taxonomy," *J. Discrete Math. Sci. Cryptogr.*, vol. 26, no. 3, pp. 677-683 (2023), doi: 10.47974/JDMSC-1740.

[25] R. Welekar, F. Shaikh, A. Chitre, K. Wanjale, S. Pathan, and A. Kumar, "An advanced cloud based framework for privacy and security in medical data using encrypted erasure-coded block transmission," *J. Discrete Math. Sci. Cryptogr.,* vol. 26, no. 5, pp. 1585–1596 (2023), doi: 10.47974/JDMSC-1826.

[26] M. P. Morolong, F. B. Shava, and A. M. Gamundani, "Cloud computing security in health cyber physical systems," *J. Discrete Math. Sci. Cryptogr.,* vol. 26, no. 5, pp. 1553–1568 (2023), doi: 10.47974/JDMSC-1821.

[27] S. Rai, A. K. Upadhyay, D. Sharma, D. Raj, A. K. Gupta, and D. Ather, "Quantum cryptography—A modern approach," *J. Discrete Math. Sci. Cryptogr.,* vol. 26, no. 7, pp. 1991–2006 (2023), doi: 10.47974/JDMSC-1839.

Received November, 2025