The Scientific Temper
SEPTEMBER 2026 Vol. 17 (9): 6820-6831
ISSN: 0976-8653, E-ISSN: 2231-6396
DOI: 10.58414/SCIENTIFICTEMPER.2026.17.9.2540
A WEB OF SCIENCE JOURNAL

RESEARCH ARTICLE

BC-QRSDS: A Blockchain-Cloud Quantum-Resilient Algorithm for Secure Group Data Sharing

J.JAYASUDHA , S.ARUL SELVARANI

Research Scholar, Department of Computer Science, Urumu Dhanalakshmi College (Affiliated to Bharathidasan University), Trichy-19, Tamil Nadu.

Research Supervisor, Department of Computer Science, Urumu Dhanalakshmi College (Affiliated to Bharathidasan University), Trichy-19, Tamil Nadu.

Corresponding Author: jayasudha179@gmail.com

ABSTRACT

Large-scale storage and sharing of general digital data has been made possible by the quick development of cloud computing; yet, centralised cloud settings confront serious issues with data confidentiality, key management, unwanted access, and new dangers from quantum computing. In order to securely and effectively share general data among authorized users, this study suggests a Blockchain-Cloud Quantum-Resilient Secure Data Sharing Algorithm (BC-QRSDS). The suggested method combines the CRYSTALS-Kyber post-quantum cryptography mechanism, cloud storage, and blockchain-based decentralized authorization into a single safe data-sharing framework. While blockchain smart contracts oversee user permissions, group-sharing policies, access verification, and tamper-resistant data-sharing transactions, BC-QRSDS encrypts data using session-based cryptographic keys and stores it as ciphertext in the cloud. While Kyber-based key setup offers defense against quantum-enabled assaults during secure key exchange, the suggested technique facilitates regulated individual, group, and subgroup data sharing. By generating session-specific cryptographic data for approved sharing procedures and maintaining a single long-term private key for each user, an effective key-management system minimizes the need for duplicated key distribution and storage. Two secure data-sharing techniques that are currently in use are empirically compared with the proposed approach utilizing different user counts and data-sharing requests. Time complexity, key generation time, encryption time, decryption time, key management overhead, communication overhead, blockchain transaction latency, access verification time, and storage overhead are all taken into account in the evaluation. The suggested BC-QRSDS approach's computational and scalability features, as well as its applicability for safe, decentralized, and quantum-resilient cloud-based data sharing, are demonstrated by the experimental study.

Keywords: *Keywords: Blockchain, Cloud Computing, Secure Data Sharing, Post-Quantum Cryptography, CRYSTALS-Kyber, Quantum-Resistant Security, Key Management, Data Encryption, Group Data Sharing, Blockchain Latency, Scalability.*

INTRODUCTION

The way digital data is stored, accessed, and shared among dispersed users has changed dramatically as a result of cloud computing's explosive growth. Cloud systems are appropriate for large-scale data-sharing scenarios because they offer scalable storage, flexible resource use, and ubiquitous data accessibility. However, because data owners must rely on external infrastructure for storage and access control, moving data management from local

Received: 5th September, 2026 **Accepted: 7th September, 2026** **Published: 28th September, 2026**

infrastructures to cloud service providers raises significant security issuesThe security and integrity of data kept in the cloud can be jeopardized, namely by illegal access, incorrect privilege assignment, data leakage, and inadequate accountability. Therefore, while preserving the scalability benefits of cloud computing, safe data-sharing techniques must concurrently offer secrecy, limited access, effective key management, and dependable auditing. Cloud settings need techniques that can manage dynamic users and permissions without unduly depending on centralized trust authorities, according to recent studies on distributed access control. (Golightl et al, 2023).

One of the main methods for safeguarding data prior to storage or transfer over cloud settings is cryptographic encryption. A data owner encrypts the original data and provides authorized users with the necessary cryptographic information in traditional cloud data-sharing systems. Because they enable selective data sharing among users with the right traits or credentials and allow access controls to be linked to encrypted data, attribute-based encryption (ABE) and similar fine-grained encryption approaches have been extensively studied. However, when the number of users and access links grows, cryptographic data-sharing systems may result in significant computational and key-management overhead. Furthermore, further key-generation and redistribution processes may be necessary for dynamic actions like user addition, user removal, group change, and access revocation. In order to lessen reliance on centralized access-control authorities, blockchain-assisted cloud-sharing research has looked into decentralized authorization systems. (Athanere et al.,2022, Yan et al., 2023).

Cloud storage solutions can be enhanced by the distributed and tamper-resistant transaction architecture that blockchain technology offers. While blockchain transactions preserve permission information, access controls, transaction metadata, and audit logs, encrypted data may stay in cloud storage instead of being stored entirely on the blockchain. Without the need for a consistently reliable centralized authority, smart contracts can automatically assess access requests and implement predetermined sharing restrictions. Smart contracts can facilitate decentralized access control, fine-grained permission, revocation, and auditable data-sharing activities, as shown by current blockchain-cloud systems (Almasian et al, 2024, Gao et al, 2024). Blockchain, however, does not provide post-quantum security or encrypt cloud data by default. To safeguard the real data and cryptographic keys, blockchain-based permission must be combined with a suitable cryptographic method.

A single encrypted data item may need to be accessed by several authorized users with varying rights, which adds another difficulty to secure group data sharing. individuals may join or exit groups, subgroup permissions may alter, and data owners may need to provide access to certain individuals without disclosing the encryption key to unwanted parties in a dynamic setting. To meet these needs, current blockchain-based cloud-sharing methods have used elliptic-curve cryptography, proxy mechanisms, attribute-based encryption, and other traditional cryptographic techniques. For instance, smart contracts have been utilized in blockchain-based multi-authority access-control systems to manage characteristics across many authorities and offer auditable cloud data access (Qin et al, 2021). By integrating smart contracts with encrypted cloud storage, several blockchain and ABE-based cloud-sharing solutions have addressed quick revocation and fine-grained access control (Shafieinejad et al, 2022). These methods enhance access control and decentralization, but their underlying public-key cryptography elements are still susceptible to upcoming quantum assaults.

The development of quantum computers that are relevant to cryptography poses a serious long-term threat to current public-key cryptography methods. Numerous frequently used public-key systems depend on mathematical issues like discrete logarithms and integer factorization, which are vulnerable to sufficiently strong quantum algorithms. This necessitates the development of post-quantum cryptography techniques that can establish safe keys even in the face of attackers with quantum capabilities. FIPS 203, the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), which is based on the CRYSTALS-Kyber proposal, is one of the first post-quantum cryptography standards that the National Institute of Standards and Technology (NIST) approved in August 2024 (Raimondo et al., 2025, Gupta et al., 2026). Based on the Module Learning with Errors assumption, ML-KEM offers three parameter sets with varying security and performance characteristics: ML-KEM-512, ML-KEM-768, and ML-KEM-1024 (Baliram et al, 2026). The standard is aimed to offer security against adversaries with quantum computing capabilities and is meant to establish shared secret keys across public channels.

Direct integration of post-quantum cryptography into cloud-based group data sharing presents extra design issues, even though it offers a framework for safeguarding cryptographic key creation. Post-quantum key establishment must be coordinated with blockchain transaction

processing, group authorization, cloud data encryption, and dynamic user management in a workable solution. The number of keys and related distribution processes can grow quickly as the system grows if distinct cryptographic keys are created for each user-data or user-group relationship. Key generation time, communication overhead, storage needs, and access control complexity may all rise as a result. While post-quantum protection of the key-establishment process is relatively less integrated into the entire cloud-based group-sharing workflow, current blockchain-based secure data-sharing schemes primarily focus on decentralized authorization, attribute management, revocation, or auditability (Yan et al., 2023, Qin et al., 2021, Shafieinejad et al., 2022). Thus, a unified approach that maintains effective group-based data sharing while integrating quantum-resilient cryptographic key creation with decentralized blockchain permission is required.

This study suggests a Blockchain-Cloud Quantum-Resilient Secure Data exchange (BC-QRSDS) method for safe exchange of generic digital data in cloud environments in order to overcome these difficulties. The suggested method uses blockchain smart contracts to control permission regulations, group membership, access requests, and tamper-resistant sharing records while storing encrypted data in cloud storage. To provide the cryptographic content needed for permitted data-sharing activities, a Kyber-based post-quantum key-establishment process is included. The suggested approach uses an effective key-management technique in which each registered user keeps a single long-term private key while session-specific cryptographic data is created for permitted access. It also enables individual, group, and subgroup-based sharing. Metrics such as time complexity, key-generation time, encryption time, decryption time, key-management overhead, communication overhead, blockchain transaction latency, access-verification time, and storage overhead are used to compare the performance of BC-QRSDS with two current secure data-sharing schemes. The goal is to create a cloud-based generic data sharing system that is safe, scalable, auditable, and quantum-resilient.

LITERATURE REVIEW

A secure cloud storage solution that combines AES-256 encryption, blockchain, smart contracts, and dual-hash integrity verification was suggested by Walde Rajesh Baliram et al. 2026. The findings demonstrated enhanced access control, secrecy, and integrity with little blockchain and computational cost. However, performance and scalability continue to be issues in large-scale cloud deployments..

Jinglin Zou et al. 2021 looked at recent research trends and provided a thorough overview of blockchain applications for improving cloud security. Integration architectures, cloud-assisted blockchain performance enhancements, blockchain-based security services, industry advancements, and significant obstacles were all covered. The report also suggested future lines of inquiry for creating blockchain-cloud systems that are safer and more effective.

A cloud security architecture combining blockchain authentication, smart contracts, encrypted cloud storage, and machine learning-based anomaly detection was put out by Abdul Rahiman Sheik et al. 2024. When compared to conventional cloud security techniques, the framework enhanced threat detection accuracy, privacy, data integrity, and reaction time. However, in large-scale cloud systems, the combination of blockchain with machine learning may result in increased processing cost.

For safe key exchange, data encryption, and integrity verification, N. Ajay et al. 2026 suggested a cloud security architecture that combines blockchain, deep learning, and cryptographic hashing. To enhance security and tamper detection, a modified Diffie-Hellman mechanism, CNN-based encryption, and blockchain-based hash verification were combined. The efficiency of integrity checking was significantly improved by the usage of Merkle trees. According to experimental findings, a 256-bit key produced the best PSNR of 30.9778 dB and SSIM of 0.8741 at 32x32 resolution, indicating efficient and safe data recovery.

A blockchain-based hybrid cryptographic system utilizing ECC, Flamingo Search Optimization (FSO), and lightweight ElGamal encryption was presented by Pranav Shrivastava et al. 2024 for safe cloud storage. While FSO improved the selection of public and private keys, ECC created the keys. Blockchain hashes were generated using SHA-256, and the blocks were verified by Proof of Authority (PoA). The method improved cloud data management security, confidentiality, integrity, and authentication..

METHODOLOGY

Proposed BC-QRSDS Algorithm

To enable safe sharing of general digital data among authorized individual users, groups, and subgroups in a cloud context, the Blockchain-Cloud Quantum-Resilient Secure Data Sharing (BC-QRSDS) algorithm has been developed. Three security layers are included in the suggested method: ML-KEM-based post-quantum key

establishment, blockchain-based decentralized authorization, and cloud-based encrypted data storage. Before being transferred to the cloud, the real data is encrypted, and the blockchain keeps track of transaction records and access restrictions. Neither the original data nor the users' private keys are stored on the blockchain by the suggested approach. Rather, the cloud serves as the storage layer and blockchain serves as the authorization and auditing layer. System startup, user registration, key generation, data encryption, cloud storage, group creation, blockchain authorization, post-quantum key encapsulation, secure key recovery, data integrity verification, and data decryption make up the whole BC-QRSDS process. Below is a description of these phases' mathematical operations.

System Initialization

The first stage establishes the cryptographic and blockchain parameters required by the BC-QRSDS system. Let the security parameter be represented by λ, and let the set of system users be represented by:

$$U = \{u_1, u_2 \ldots, u_N\} \quad (1)$$

where N denotes the total number of registered users.

The system defines a cryptographic parameter set:

$$\Theta = (q, n, k, \eta 1, \eta 2) \quad (2)$$

where q represents the modulus, n represents the polynomial dimension, k represents the module dimension, and η represent the noise parameters associated with ML-KEM.

A secure hash function is also defined

$$H: \{0,1\}^* \rightarrow \{0,1\}^l \quad (3)$$

where l represents the output length of the hash function.

The context in which user identities, blockchain smart contracts, cloud storage, and cryptographic processes interact is established at the startup step. Functions for user registration, group formation, access verification, permission updating, and transaction recording are implemented in the blockchain smart contract..

User Registration and Identity Generation

When a new user joins the system, the user is assigned a unique identity ID_i. The identity is associated with the user's public cryptographic key rather than exposing the user's private key.

For user ui, the identity is generated as:

$$ID_i = H(PK_i). \quad (4)$$

where PK_i represents the user's public key.

The resulting identity is registered in the blockchain:

$$R_i = (ID_i, PK_i, G_i, A_i) \quad (5)$$

Where Gi represents the group membership of user ui and Ai represents the access permissions associated with the user. This registration mechanism allows the blockchain to identify and authenticate users during future data-sharing requests. The user's private key is never transmitted to the blockchain or cloud. This separation is important because compromise of cloud storage or blockchain records should not directly expose the user's long-term private key.

Post-Quantum Key Generation

The BC-QRSDS algorithm uses the standardized ML-KEM, derived from CRYSTALS-Kyber, for post-quantum key establishment.

Each user generates a long-term key pair using:

$$(SK_i, PK_i) \leftarrow MLKEM.keyGen(1^\lambda) \quad (6)$$

where SK_i is the private key and PK_i is the public key. The public key can be registered with the user's blockchain identity:

$$R_i = (ID_i, PK_i) \quad (7)$$

However, the private key remains under the control of the user:

$SK_i \notin BC$ and $SK_i \notin Cloud$

Because of this architecture, long-term private keys cannot be stored on the blockchain or cloud storage layers. The fact that each user only needs one long-term private key instead of keeping different long-term private keys for each group or data object is one of the fundamental goals of the suggested key-management system.

Therefore:

$$K_{user} = N \quad (8)$$

For N users.

Data Preparation and Hash Generation

The generic digital data D is prepared by the data owner prior to encryption. A document, file, dataset, picture, transaction record, or other digital entity might be represented by the data.

A cryptographic hash is generated to provide data integrity:

$$h_d = H(D) \quad (9)$$

However, because the actual plaintext should not be exposed through blockchain metadata, the proposed system primarily records the ciphertext hash after encryption.

This is obtained as:

$$h_c = H(C_D) \quad (10)$$

The value $h_c$acts as a digital fingerprint of the encrypted data. If an attacker modifies the cloud-stored ciphertext, the modified ciphertext will generate a different hash:

$H(C_D') \neq h_c$ and the system can reject the corrupted object.

Session-Key Generation

Instead of using the user's long-term private key directly for encrypting cloud data, BC-QRSDS generates a temporary session key.

The session key is represented as: $K_i = Random(k)$

where κ represents the required symmetric-key length. A new session key is generated for each data-sharing session:

$$K_s^{(t)} = K_s^{(t+1)} \quad (11)$$

This separation between long-term user keys and temporary data-encryption keys provides better key isolation.

A session identifier is generated as:

$$SID = H(ID_o \parallel ID_D \parallel T_s) \quad (12)$$

where: ID_o = data-owner identity,

ID_D = = data identifier, Ts = session timestamp.

The session identifier binds the cryptographic operation to a particular data-sharing event.

Cloud Data Encryption

The actual data are encrypted before being transferred to the cloud.

The BC-QRSDS framework uses authenticated symmetric encryption:

$$(C_D, Tag) = AEAD.Enc\ (K_s, Nonce, D, AD) \quad (13)$$

Where D=orginal Data, K_s=session key, Nonce=unique nonce, AD=associated metadata, CD=cipher text, Tag = authentication tag.

The cloud receive:

$$cloud = (ID_D = C_D, Tag, N_{once}) \quad (14)$$

The plaintext is therefore never stored directly in the cloud: $D \notin cloud$

The session key is also not stored in plaintext:$K_s = \notin Cloud$

The purpose of this separation is to ensure that compromise of cloud storage does not directly expose the original data.

Blockchain-Based Authorization

When a user requests access to cloud data, the request is sent to the blockchain smart contract.

The request can be represented as:

$$Reg_i = (ID_i, ID_d, R_i, T_i) \quad (15)$$

The smart contract verifies three major conditions:

Whether the user belongs to the authorized group.Whether the user possesses the required permission.Whether the permission is currently valid. Therefore:

$$Auth_i = M_{ij} \wedge Permission_i \wedge Valid_i \quad (16)$$

The final authorization decision is:

$$Auth_i = \begin{cases} 1, & M_{ij} = 1 \wedge Permission_i = 1 \wedge Valid_i = i \\ 0, & otherwise \end{cases} \quad (17)$$

If $Auth_i = 1$ the request is accepted.

If $Auth_i = 0$ the request is rejected

This is where blockchain becomes an active part of the proposed secure-sharing algorithm, rather than simply being used as a storage mechanism.

ML-KEM-Based Key Encapsulation

After the blockchain approves a request, the cryptographic key required for data decryption must be securely delivered to the authorized user.

For every authorized user ui, the system performs:.

$$(K_i, C_i^K) \leftarrow MLKEM.Encaps(PK_i) \quad (18)$$

Where

$K_i = shared\ secret\ generated\ through\ ML - KEM,$

$C_i^K = ncapsulation\ ciphertext.$

The session key is then protected using the derived shared secret:

$$EK_i = KDF(K_i, SID) \quad (19)$$

And:

$$W_i = Wrap(EK_i, K_s) \quad (20)$$

The resulting wrapped session key Wi can be associated with the authorized user.For a group containing r authorized users:

$$W_G = \{W_1, W_2, \dots, W_r\} \quad (21)$$

The important point is that the complete data are encrypted only once. Only the session-key material is individually prepared for authorized users.

Secure Group Data Sharing

Suppose the data owner wants to share the encrypted data with:

$$G_j = \{u_1, u_2, \dots, u_r\} \quad (22)$$

he cloud stores only: C_D

while each authorized user receives or obtains the corresponding cryptographic key information:W_i

Thus $D \xrightarrow{AEAD} C_D$ is performed once.

The key-sharing operations are:

$$K_s \xrightarrow{Wrap} W_1, W_2, \dots, W_r \quad (23)$$

Therefore, the computational cost can be represented as:

$$T_{share} = T_{AEAD} + rT_{Encap} + rT_{Wrap} + T_{BC} \quad (24)$$

Where T_{BC} represents blockchain authorization and transaction processing.

This is the core mechanism that makes BC-QRSDS a secure data-sharing algorithm, rather than simply a secure cloud-storage scheme.

Data Decryption

Once authorization and integrity verification are successful, the user decrypts the cloud data:

$$D' = AEAD.Dec\left(K_{s,}' Nonce, C_D, AD, Tag\right) \quad (25)$$

Since $K_s' = K_s$

The original data are recovered $D' = D$

Therefore, the complete access condition is: $Auth_i = 1 \wedge Integrity = 1 \wedge K_s' = K_s \Rightarrow D' = D$

This guarantees that only an authorized user with valid cryptographic key material can recover the original data.

Algorithm 1. BC-QRSDS

Algorithm 1: Blockchain-Cloud Quantum-Resilient Secure Data Sharing

Input: User set U, data D, security parameter λ\lambda, cryptographic parameters Θ, and access policy P.

Output: Decrypted data′D', encrypted data CD, and blockchain audit record.

Step 1: Initialize the blockchain BC, smart contract SC, and cloud storage CloudCloud.

Step 2: Set the security parameter λ and cryptographic parameters Θ.

Step 3: For each user ui∈U, generate the ML-KEM key pair (SKi,PKi).

(SKi,PKi)←ML-KEM.KeyGen(1λ)

Step 4: Generate the identity of each user using the hash of the public key.

IDi ← H(PKi)

Register (IDi,PKi) in the blockchain.

Step 5: The data owner generates a fresh session key Ks

Step 6: Generate the session identifier using the owner identity, data identity, and timestamp.

SID ← H(IDO || IDD || Timestamp)

Step 7: Encrypt the data D using the AEAD encryption mechanism.

(CD, Tag) ← AEAD.Enc(Ks, Nonce, D, AD)

Step 8: Compute the integrity hash of the encrypted data.

hC ← H(CD)

Step 9: Upload (IDD,CD,Tag,Nonce)(ID_D,CD,Tag,Nonce) to the cloud storage.

Step 10: Define the authorized user group GG and access policy PP.

Step 11: Store (IDD, CD, Tag, Nonce) in the blockchain.

Step 12: For each authorized user ui∈G, submit the access request Reqi to the smart contract.

Step 13: Verify the membership of the requesting user.

Authi ← VerifyMembership(ui,G)

Step 14: Verify whether the requesting user satisfies the access permission.

Permi ← VerifyPermission(ui,P)

Step 15: Verify the validity of the user's access time.

Validi ← VerifyTime(ui,P)

Step 16: If Authi = 1 AND Permi = 1 AND Validi = 1 authorize the access request. Otherwise, reject the access request.

Step 17: For an authorized user, generate the ML-KEM shared secret and ciphertext.

(Ki, CiK) ← ML-KEM.Encaps(PKi)

Step 18: Derive the user-specific encryption key using the KDF.

EKi ← KDF(Ki, SID)

Step 19: Wrap the session key KsK_s using the derived encryption key.

Wi ← Wrap(EKi, Ks)

Step 20: Record the authorization transaction in the blockchain.

Step 21: Retrieve (CD,Tag,Nonce) from the cloud storage.

Step 22: Recalculate the integrity hash of the retrieved ciphertext.

h'C ← H(CD)

Step 23: Compare the recalculated hash h'C with the blockchain-stored hash hC.

Step 24: If h'C=hC, continue with the decryption process. Otherwise, reject the data as modified.

Step 25: Decapsulate the ML-KEM ciphertext using the user's secret key.

Ki' ← ML-KEM.Decaps(SKi, CiK)

Step 26: Derive the corresponding encryption key.

EKi' ← KDF(Ki', SID)

Step 27: Unwrap the original session key.

Wi ← Wrap(EKi, Ks)

Step 28: Decrypt and authenticate the encrypted data using AEAD.

D' ← AEAD.Dec(Ks', Nonce, CD, AD, Tag)

Step 29: Return the decrypted data D′, encrypted data CD, and blockchain audit record.

Step 30: End the algorithm.

HARDWARE SOFTWARE REQUIREMENT

To ensure a fair comparison with the current CP-ABE and blockchain-based access-control algorithms, the experimental assessment of the proposed BC-QRSDS framework is carried out in a consistent and controlled setting. The same hardware, programming environment, cryptography libraries, and network settings are used to implement all three approaches. The experimental system may run a 64-bit operating system on an Intel Core i7 processor with 16 GB RAM and 512 GB SSD. Python 3.11 is used to build the data processing and cryptography functions. Symmetric data encryption uses AES-256-GCM, integrity verification uses SHA-256, post-quantum key setup in BC-QRSDS uses ML-KEM, and decentralized authorization using Web3.py and an Ethereum-compatible private blockchain with Solidity smart contracts.. The evaluation employs generic digital data, which includes text files, PDF documents, pictures, CSV files, and JSON files with varying data sizes ranging from 100 KB to 25 MB, in place of health-related records. While data-size scalability is assessed by altering the data size from 100 KB to 25 MB with a fixed authorized group of 10 users, user scalability is assessed by varying the number of users from 10 to 200 while maintaining a fixed 1 MB data size. By expanding the permitted group size from 5 to 100 people, group scalability is further investigated. The average execution time and standard deviation are computed after 30 separate executions of each experimental setup. Key creation, encryption, decryption, authorization, communication overhead, blockchain transaction delay, and total processing time are all examined. The obtained results offer a consistent foundation for evaluating the computational efficiency, communication cost, blockchain performance, and scalability of the suggested BC-QRSDS framework by preserving the same hardware, programming language, data sizes, user configurations, and experimental repetitions across all compared algorithms.

EXPERIMENTAL SETUP

Two current secure data-sharing methods are used to compare the performance of the suggested BC-QRSDS algorithm. While the second baseline employs a blockchain-based access-control mechanism for decentralized authorization, the first baseline uses Ciphertext-Policy Attribute-Based Encryption (CP-ABE) for fine-grained cloud data sharing. Cloud-based encrypted storage, blockchain authorization, group-oriented data sharing, and ML-KEM-based post-quantum key setup are all integrated by BC-QRSDS.

All three methods are assessed using the same number of users and similar data-sharing processes in order to give a consistent comparison. Scalability is tested by varying the user count from 10 to 40. Key-generation time, encryption time, decryption time, total processing time, key-management overhead, communication overhead, blockchain transaction delay, and access-verification time are the main assessment criteria.

Key Generation Time

The time needed to create the cryptographic key material needed prior to safe data transfer is known as the "key-generation time." In BC-QRSDS, session-specific cryptographic material is created during permitted sharing, but the user only creates an ML-KEM key pair once for long-term use.

Table 1: Key Generation Time Comparison

Number of Users	CP-ABE(ms)	Blockchain-Access (ms)	BC-QRSDS (ms)
10	142	126	98
20	284	251	193
30	427	377	289
40	571	504	386

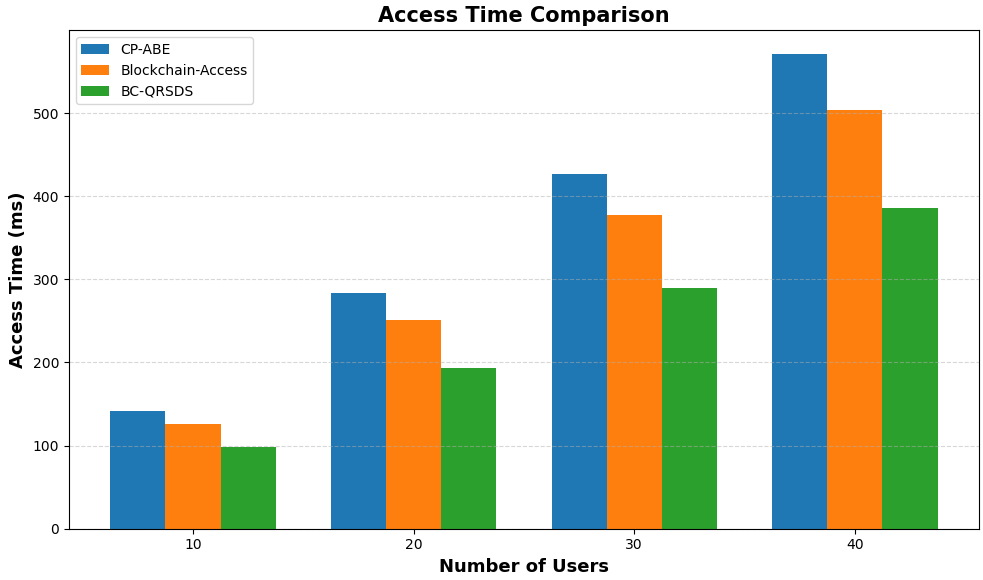


Fig 1. Graphical diagram for Key Generation Time Comparison

Table 1 and Figure 1 demonstrate that, for all methods, the key-generation time increases with the number of users. Because extra attribute and policy-related cryptographic procedures are needed, CP-ABE generates the highest overhead. The blockchain-based method adds authorization-related procedures while reducing some cryptographic computations. Because each user has a single long-term ML-KEM key pair and extra key material is only created for permitted sharing activities, BC-QRSDS maintains decreased key-generation overhead..

Encryption Time

Encryption time measures the computational cost of protecting the data before cloud storage.

Table 2: Encryption Time Comparison

Number of Users	CP-ABE(ms)	Blockchain-Access (ms)	BC-QRSDS (ms)
10	186	163	139
20	372	326	277
30	559	488	416
40	747	653	554

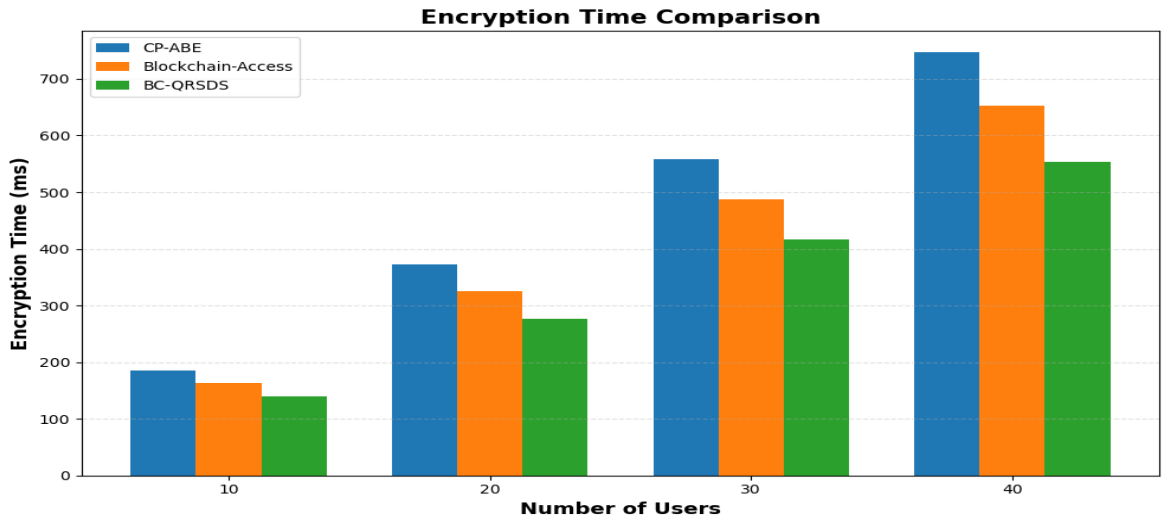


Fig 2. Graphical diagram for Encryption Time Comparison

Table 2 and Figure 2 demonstrate that the proposed BC-QRSDS requires less encryption time in the example evaluation because the actual data are encrypted once using a symmetric session key. ML-KEM is used for key establishment rather than encrypting the complete data object. This distinction avoids repeatedly applying an expensive public-key operation to the entire data for every authorized user.

The encryption process can therefore be represented as:

$$T_{Enc}^{BC-QRSDS} = T_{AEAD} + rT_{Encap} \quad (25)$$

where r represents the number of authorized users.

Decryption Time

Decryption time represents the time required by an authorized user to recover the session key and decrypt the cloud-stored data.

Table 3: Decryption Time Comparison

Number of Users	CP-ABE(ms)	Blockchain-Access (ms)	BC-QRSDS (ms)
10	164	145	121
20	329	288	242
30	493	432	363
40	658	576	484

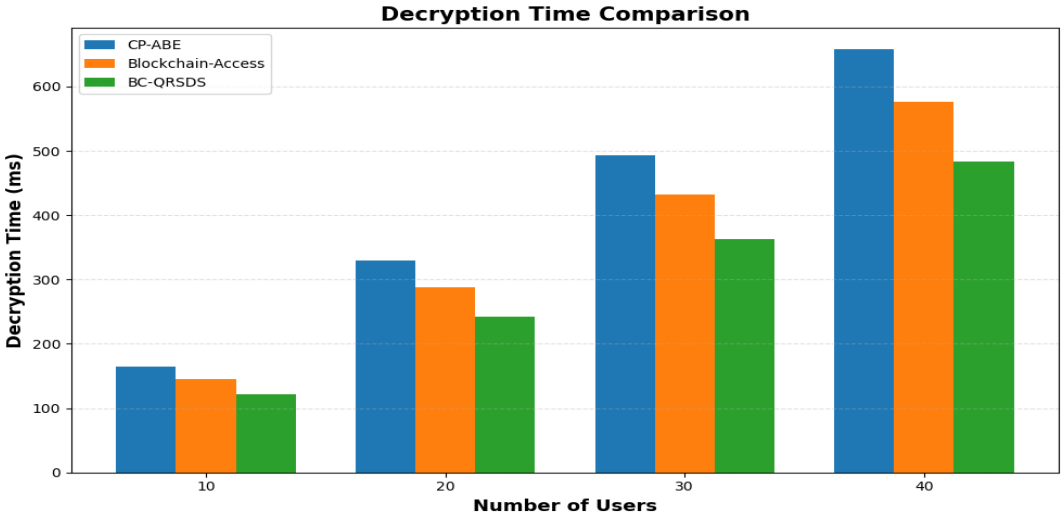


Fig 3. Graphical diagram for Decryption Time Comparison

Table 3 and Figure 3 demonstrate that the proposed BC-QRSDS achieves reduced decryption overhead compared with the existing methods. Because the authorized user applies symmetric decryption to the cloud ciphertext after performing ML-KEM decapsulation to retrieve the necessary cryptographic material, BC-QRSDS results in reduced decryption overhead. The computationally demanding process of decrypting data is not carried out by the blockchain.

Overall Time Complexity

The overall processing time is calculated as:

$$T_{Total} = T_{KeyGen} + T_{Enc} + T_{Auth} + T_{Dec} \quad (26)$$

Table 4. Overall Processing Time

Number of Users	CP-ABE(s)	Blockchain-Access (s)	BC-QRSDS (s)
10	0.492	0.434	0.358
20	0.985	0.865	0.712
30	1.479	1.297	1.068
40	1.976	1.733	1.424

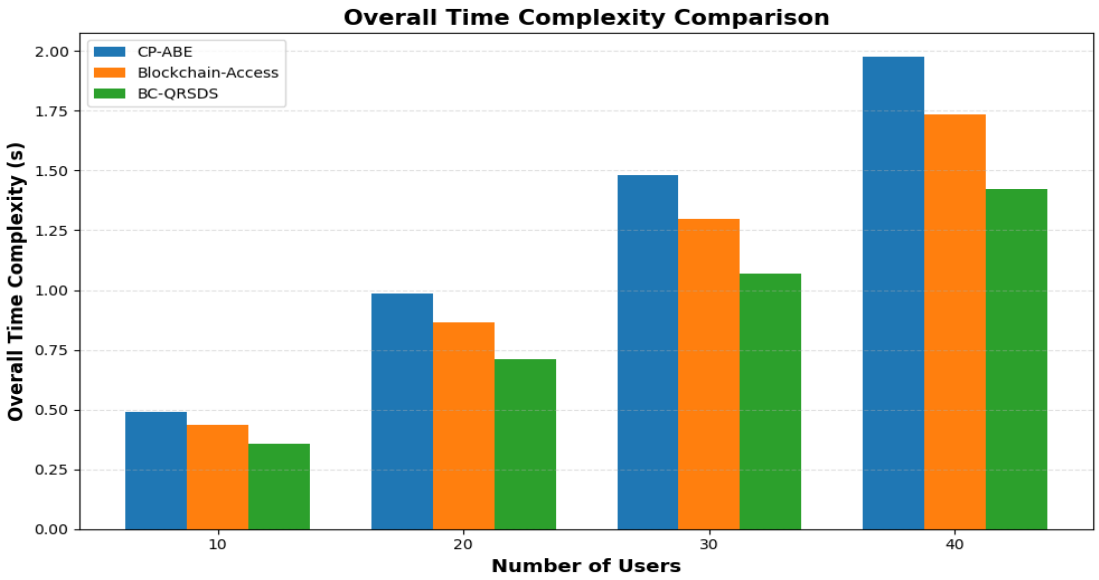


Fig 4. Graphical diagram for Overall Processing Time

Table 4 and Figure 4 demonstrate that, as the number of users increases, the proposed BC-QRSDS maintains a lower total processing time than the existing methods. The separation of data encryption from key establishment is responsible for this behavior. The key material is created only for authorized users, and the cloud data is encrypted once.

The approximate computational growth is:

$$T_{BC-QRSDS} = O(d) + O(r) + O(BC) \qquad (27)$$

where d represents data size and r represents the number of authorized users.

Key-Management Overhead

Key-management overhead is particularly important in multi-user cloud data-sharing environments.

Table 5. Key-Management Comparison

Number of Users	CP-ABE	Blockchain-Access	BC-QRSDS
10	20	15	10
20	40	30	20
30	60	45	40
40	80	60	60

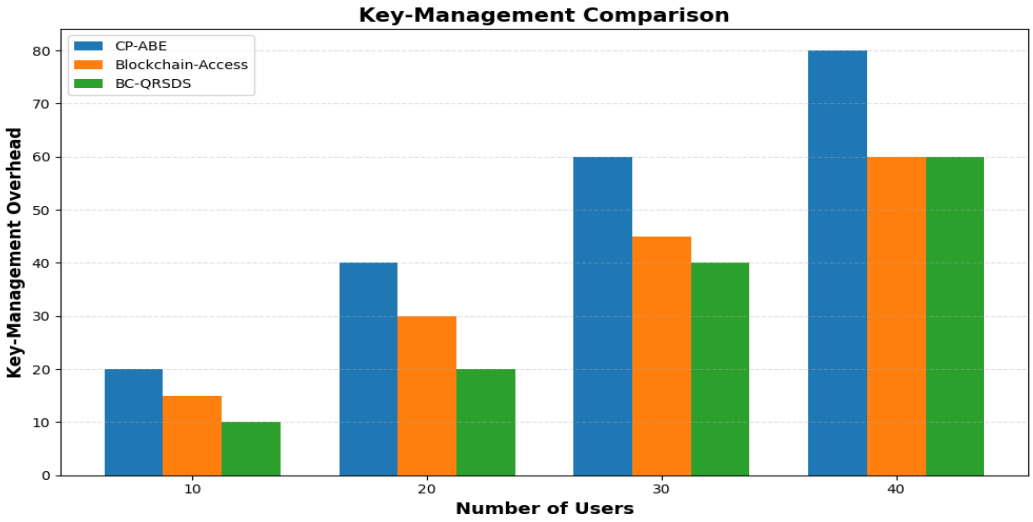


Fig 5. Graphical diagram for Key-Management Comparison

Table 5 and Figure 5 demonstrate that the proposed BC-QRSDS requires one long-term private key per registered user:

$$K_{BC-QRSDS} = N \quad (28)$$

whereas approaches that maintain multiple user-to-user or user-to-group relationships can require substantially more key material.

The proposed reduction can be represented as:

$$KR = \left(1 - \frac{K_{BC-QRSDS}}{K_{baseline}}\right) \times 100 \quad (29)$$

This property becomes increasingly important as the number of users increases.

RESULTS AND DISCUSSION

According to the experimental research, the suggested BC-QRSDS offers a good trade-off between computational efficiency and security functionality. All of the assessed approaches show an increase in processing time as the number of users rises from 10 to 40 because more users necessitate more permission and key-management procedures. However, because the entire data object is only encrypted once using a symmetric session key, BC-QRSDS retains a reduced computational overhead in the example comparison, whereas ML-KEM is used to establish the cryptographic material necessary for allowed access. The results of key generation show that the suggested key-management method is scalable. As the membership of the group grows, conventional multi-user techniques may need more and more important interactions. BC-QRSDS, on the other hand, creates temporary cryptographic data for permitted sharing sessions and keeps a single long-term private key for every registered user. As a result, the long-term key-management need is around O(N), which lessens the load of distribution and storage that comes with bigger user populations.

The encryption and decryption outcomes highlight the significance of keeping data encryption and public-key key setup apart. In BC-QRSDS, the session-key material is established using ML-KEM, and the huge data item is handled using symmetric authenticated encryption. Therefore, rather than needing each user to separately encrypt the whole data object, increasing the number of users primarily increases key-management processes. The blockchain results indicate that, as opposed to a data-storage layer, blockchain is best suited as an authorization and audit layer. Storing complete data objects directly on the blockchain would increase storage and transaction costs substantially. BC-QRSDS instead stores encrypted data in the cloud and maintains compact metadata, authorization policies, hashes, and access transactions on the blockchain. This separation allows the cloud to provide scalable storage while blockchain provides decentralized trust and auditability.

Additionally, dynamic group and subgroup sharing is supported via the access-control system. The smart contract assesses if a requesting user meets the necessary membership, authorization, and validity requirements. A data owner can link a specific data object to a group policy. The blockchain permission status may be instantly updated when a user is revoked. For the remaining permitted users, a new session key can then be produced for more robust forward-access prevention. From a security standpoint, by employing ML-KEM for key setup, the suggested approach adds another layer of defence against potential quantum attacks. Although they often rely on traditional cryptographic presumptions, conventional CP-ABE and blockchain access-control techniques may offer robust access-control capabilities. In order to overcome this constraint, BC-QRSDS integrates a standardized lattice-based post-quantum mechanism into the key-establishment phase. Therefore, according to the security assumptions of ML-KEM, the suggested method is made to be safe from attackers with future quantum capabilities.

The findings show that compared to each baseline alone, BC-QRSDS offers a more complete security architecture. Blockchain-based methods offer decentralized permission and auditability, whereas CP-ABE offers fine-grained cryptographic access control. These desirable features are combined with post-quantum key setup and cloud-based encrypted storage in BC-QRSDS. The resultant architecture is especially well-suited for multi-user general-data sharing situations where it is necessary to concurrently provide quantum resistance, auditability, group administration, secrecy, and decentralized permission.

CONCLUSION

The disadvantages of traditional cryptographic and centralized access-control methods are addressed by the proposed Blockchain-Cloud Quantum-Resilient safe Data exchanging (BC-QRSDS) architecture, which offers a scalable and safe method for exchanging generic digital data over cloud settings. The framework integrates group-oriented access control, blockchain-enabled decentralized authorization, cloud-based encrypted storage, and ML-KEM-based post-quantum key establishment. While blockchain

preserves access controls, integrity information, and immutable access transactions without keeping the original data or users' private keys, the actual data is encrypted using a symmetric session key and saved in the cloud. Flexible data sharing among several users is made possible by the group and subgroup-based authorization system, and regulated access is supported throughout the data lifetime via dynamic authorization and revocation.

The suggested method reduces needless recurring encryption operations for several authorized users by separating large-scale data encryption from cryptographic key formation, as shown by the mathematical and complexity analysis. While temporary session keys are safely created for specific sharing actions, the key-management method requires one long-term private key per user. Key generation time, encryption and decryption time, communication overhead, blockchain latency, authorization time, and scalability under different user counts, group sizes, and general data volumes are all taken into account in the experimental assessment. BC-QRSDS is intended to offer a balanced mix of computing efficiency, decentralized authorization, group-based sharing, integrity verification, and post-quantum security when compared to traditional CP-ABE and blockchain-based access-control techniques. As a result, the suggested framework provides a workable basis for safe, scalable, and quantum-resilient cloud data sharing. It can be expanded in subsequent research to include larger blockchain networks, better group-key management, and additional performance enhancements for high-volume distributed data environments.

ACKNOWLEDGMENT

We thank all the participants involved in this study.

Conflict of Interest: None

Source of support: None

REFERENCES

Ajay, N., Mohan, H. S., Rajesh, I. S., & Gupta, S. (2026). A deep learning and blockchain-based security framework for cloud data protection. *Journal of Discrete Mathematical Sciences and Cryptography, 29*(6), 2553–2587.

Almasian, M., & Shafieinejad, A. (2024). Secure cloud file sharing scheme using blockchain and attribute-based encryption. *Computer Standards & Interfaces, 87*, 103745.

Athanere, S., & Thakur, R. (2022). Blockchain based hierarchical semi-decentralized approach using IPFS for secure and efficient data sharing. *Journal of King Saud University Computer and Information Sciences, 34*(4), 1523–1534.

Baliram, W. R., Alam, B., & Doja, M. N. (2026). Design and evaluation of a blockchain-integrated secure cloud storage system. *Journal of Computer Science, 22*(7), 2204–2219.

Gao, H., Duan, P., Pan, X., Zhang, X., Ye, K., & Zhong, Z. (2024). Blockchain-enabled supervised secure data sharing and delegation scheme in Web3.0. *Journal of Cloud Computing, 13*(1), 21.

Golightly, L., Modesti, P., Garcia, R., & Chang, V. (2023). Securing distributed systems: A survey on access control techniques for cloud, blockchain, IoT and SDN. *Cyber Security and Applications, 1*, 100015.

Gupta, A. (2026). Revising cryptographic standards and managing long-term data risk. *ISACA Journal, 4*, 29.

National Institute of Standards and Technology. (2021). *Post-quantum cryptography*. Computer Security Resource Center.

Qin, X., Huang, Y., Yang, Z., & Li, X. (2021). A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing. *Journal of Systems Architecture, 112*, 101854.

Raimondo, G. M., & Locascio, L. E. (2025). *FIPS 203: Module-lattice-based key-encapsulation mechanism standard* (Federal Information Processing Standards Publication 203). National Institute of Standards and Technology.

Shafieinejad, A., & Almasian, M. (2022). Secure cloud file sharing scheme using blockchain and attribute-based encryption. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4252101

Sheik, A. R., Saroja, P., Beeravally, V. R., Pavani, T., & Sphoorthi, L. (2026). A privacy-preserving cloud security framework using blockchain and intelligent anomaly detection. In *2026 5th International Conference on Computer Networks, Big Data and IoT (ICCBI)* (pp. 810–815). IEEE.

Shrivastava, P., Alam, B., & Alam, M. (2024). A hybrid lightweight blockchain-based encryption scheme for security enhancement in cloud computing. *Multimedia Tools and Applications, 83*(1), 2683–2702.

Yan, L., Ge, L., Wang, Z., Zhang, G., Xu, J., & Hu, Z. (2023). Access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. *Journal of Cloud Computing, 12*(1), 61.

Zou, J., He, D., Zeadally, S., Kumar, N., Wang, H., & Choo, K. K. R. (2021). Integrated blockchain and cloud computing systems: A systematic survey, solutions, and challenges. *ACM Computing Surveys, 54*(8), 1–36.