RESEARCH ARTICLE

INTERNATIONAL RESEARCH JOURNAL OF MULTIDISCIPLINARY TECHNOVATION

Asian Research Association

DOI: 10.54392/irjmt2558

Dynamic and Revocable Multi File-Multi User Access Control based Aggregate Key Cryptosystem to Secure Data Sharing in Cloud Computing

Sameera Mahammad [a, b, *], K. Usha Rani [a]

[a] Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati, Andhra Pradesh, India
[b] Department of Computer Science, D.K. Govt. College for Women (A), Nellore, Andhra Pradesh, India
* Corresponding Author Email: samee2016@gmail.com

DOI: https://doi.org/10.54392/irjmt2558
Received: 07-11-2024; Revised: 23-08-2025; Accepted: 09-09-2025; Published: 25-09-2025

Abstract: The emergence of cloud computing and IoT has made effective and secure cryptographic schemes essential for sharing data online. To protect sensitive data, data owners must encrypt files before storing them online and grant decryption rights to authorized users. A novel approach, the Key Aggregate Cryptosystem (KAC), enables users to decode multiple pieces of data types with one single constant key size, enhancing efficiency. In this article the Efficient Revocable & Dynamic Secure Aggregate Key Cryptosystem Approach (ERDSAKCA) is tailored for cloud environments. This innovative strategy not only simplifies the key management through KAC but also allows for dynamic user revocation by updating the ciphertext, ensuring revoked users cannot access new data while non-revoked users retain access without updating their private keys. The scheme also incorporates a verification mechanism to ensure accuracy in user revocation and ciphertext updates. Compared to existing schemes, ERDSAKCA effectively manages user access control and user revocation and reduces the costs associated with key management and storage. Lastly, the scheme is shown to be selectively chosen plaintext-safe under the conventional model, offering strong protection against cryptographic attacks.

Keywords: Aggregate Key Cryptosystem, Secure Data Sharing, Semantic Security Mechanism, Dynamic Access Control Policy, Attribute Based Encryption.

1. Introduction

Cloud computing has emerged as a transformative model that provides flexible, scalable, and cost-effective access to shared computing resources. Among its core services, cloud storage offers virtualized storage infrastructure that reduces dependency on local hardware and facilitates seamless data access and management across distributed environments. With the exponential growth in data, organizations and individuals increasingly rely on cloud-based services to store, retrieve, and share data efficiently. Cloud storage allows users to offload massive data to remote servers, improving operational agility and minimizing infrastructure costs [1, 2]. Additionally, in multi-user settings, the ability to share content securely and selectively is essential for collaborative applications in fields like healthcare, education, and finance. However, this convenience introduces critical security concerns, especially when data owners relinquish control to potentially untrusted cloud service providers (CSPs) [3].

Despite the growing adoption of cloud services, access control remains a significant challenge. Traditional access mechanisms either rely heavily on CSPs or involve complex cryptographic management that is impractical at scale. Moreover, ensuring fine-grained access control, while allowing dynamic changes (e.g., revoking users or modifying access policies), is particularly difficult in distributed environments, and many existing solutions are largely static, requiring costly re-encryption and key redistribution that can disrupt availability of services for authorized users [4].

Recent advances in key-aggregate cryptosystems and searchable encryption address revocation, delegation, and verifiability to varying degrees [5–13]. Despite these advancements, key challenges remain unaddressed**.** Many existing models lack support for simultaneous revocation, verification, and aggregation in a scalable manner, particularly in multi-user, multi-file cloud settings. Most schemes continue to rely on CSPs for executing revocation and ciphertext updates without any guarantee of verifiability from the data owner's side. This introduces trust issues

and potential for unauthorized access. Furthermore, frequent re-keying and re-encryption in these schemes cause significant overhead and limit real-time collaboration. These limitations motivate a lightweight, verifiable, and scalable revocable KAC scheme that ensures secure access control and efficient key management without affecting the accessibility of data for legitimate, non-revoked users. A structured comparison is provided in Section 2 (Table 1) and highlights the specific gaps that motivate this work.

This article proposes the Efficient Revocable & Dynamic Secure Aggregate Key Cryptosystem Approach (ERDSAKCA) with the following contributions:

- Efficient Key Management: Using KAC to aggregate multiple private keys into a single, constant-sized key.
- Dynamic User Revocation: Supports revocation by updating ciphertext, ensuring revoked users cannot access new data without requiring non-revoked users to update keys.
- Verification Mechanism: Includes a verification algorithm to confirm accurate revocation and ciphertext updates.
- Security Proof: Proven semantically secure under the Generalized Decisional Hybrid Diffie-Hellman Exponent (DHDHE) assumption.
- Performance Evaluation: Evaluates the performance of the proposed scheme with the existing schemes. The analysis shows that our scheme reduces computational and storage costs compared to existing schemes.

The subsequent sections of the article are structured as follows. Section 2 presents the tabular literature review. Section 3 outlines the materials and methods, including the system model, cryptographic background, and construction of the proposed ERDSAKCA scheme. Section 4 presents the experimental results, evaluating key performance metrics such as setup time, encryption/decryption efficiency, revocation performance, and memory usage. Section 5 provides a detailed discussion, comparing ERDSAKCA with existing schemes and highlighting its advantages. Finally, Section 6 concludes the study and suggests potential directions for future work. Author declarations, Data Availability and References are provided at the end of the manuscript.

2. Literature Review

To situate ERDSAKCA within current research, Table 1 summarizes closely related KAC/KASE and revocation oriented schemes relevant to multi-user, multi-file cloud data sharing.

3. Materials and Methods

This section describes the foundational concepts and cryptographic mechanisms employed in the development of the proposed ERDSAKCA scheme. The materials include the mathematical assumptions, group structures, and cryptographic primitives used to construct the key-aggregate framework. The methods outline the setup of the system, key generation, encryption and decryption processes, user revocation strategy, and the integrated verification mechanism. These components collectively enable secure, scalable, and flexible data sharing in a multi-user cloud environment.

3.1 Basic Preliminaries

This section, introduces the foundational concepts and definitions related to KAC and its semantic security, along with complexity assumption called the Generalized Decisional n-Hybrid Diffie-Hellman Exponent (Generalized DHDHE).

3.1.1 Key Aggregate Cryptosystem (KAC)

The KAC, a cryptographic framework for secure data sharing with fine-grained access control comprises the following six randomized algorithms [14]:

1. The Setup function initializes the system with public parameters PK and a secret parameter *t*, which is only accessible to authorized data owners.
2. The Keygen function outputs the public and master-secret key pair (PK = N̂p, msk = N̂).
3. The Encrypt function takes three parameters: a public key PK, a ciphertext class *i*, and a message *m*. It returns the class *i* ciphertext C corresponding to message *m*.
4. The UserRevoc function manages user access by taking msk and a subset of ciphertext classes S, computing access control parameters and an aggregate key KS.
5. The Access Control and Key Distribution delivers the aggregate key KS and access control parameter U to users with appropriate access rights through a secure channel.
6. The Decryption function decrypts the ciphertext C for specific class *i* using aggregate key KS and access control parameter U, and produces the decrypted message *m*.

3.1.2 Keeping KAC Secure Semantically

The semantic security of a key-aggregate encoding system in opposition to an adversary is defined through the interaction between the attack algorithm A and the challenger algorithm B.

Table 1. Summary of related work

Ref	Core idea	Revocation	Aggregation / Key Size	Verification	Scope	Gaps Addressed by ERDSAKCA
[5]	Blockchain-based revocable KASE for group sharing (IIoT)	Yes (non-interactive)	Aggregate search keys (not general KAC)	Blockchain-assisted verification	Search	Search-only focus, blockchain complexity/ latency; R DSAKCA targets file sharing with lightweight verification
[6]	KAC with user revocation for selective groups	Yes (ciphertext updates)	Constant-size ciphertexts	No explicit CSP-update check	Files	Updates may affect non-revoked users; ERDSAKCA confines impact and adds verifiability
[7]	Key-aggregate proxy re-encryption (KAPRE)	Yes (via re-encryption)	Aggregation via PRE	Not emphasized	Files	Re-encryption cost on revocation; ERDSAKCA avoids per-revocation re-encryption burden
[8]	Verifiable data sharing in dynamic multi-owner	Not core focus	Not key-aggregate	Yes (verifiable sharing)	Files	No lightweight aggregation; ERDSAKCA integrates aggregation and verification
[9]	Multi-owner KASE	Partial (search-oriented)	Search-key aggregation	Not primary	Search	Search-only; ERDSAKCA supports general file sharing and dynamic revocation
[10]	Revocable online/offline KASE (reduced online cost)	Yes (search context)	Search-key aggregation	Not primary	Search	Search-specific; ERDSAKCA generalizes to files and adds ciphertext-update verification
[11]	KAASE (Authorized searchable encryption with multi-key)	Yes (search context)	Aggregate/ authorized search keys	Not primary	Search	Preprint and search-centric; ERDSAKCA covers multi-file access and verification
[12]	Key-aggregate authentication cryptosystem	Not core	Aggregate authentication	No CSP-update check	Files	No verifiable revocation updates; ERDSAKCA includes verification
[13]	Identity Based conditional PRE with ciphertext evolution	Yes (Via evolution)	Delegation focused	Not primary	Files	Key-management complexity; ERDSAKCA keeps constant-size aggregates with simple updates
[14]	KAC-based access control for flexible sharing	*Not native; achievable with KAC variants.*	Constant-size aggregates	Not primary	Files	No built-in verification; ERDSAKCA integrates verification and dynamic updates
[15]	KASE foundations & implementation	Search-token revocation only (not general file access)	Search-key aggregation	Proofs for search result correctness; not CSP update checks.	Search	Limited to search; ERDSAKCA targets general file ciphertexts
[18]	Efficient verifiable KASE	Revocable (search)	Search-key aggregation	Yes (for search)	Search	Verification tied to search; ERDSAKCA verifies file ciphertext updates
[19]	KASE under multi-owner setting	Partial (search)	Search-key aggregation	Not primary	Search	Multi-owner search; ERDSAKCA supports multi-owner/multi-file and CSP-side checks
[20]	BTG-RKASE with fine-grained multi-delegation "break-the-glass"	Yes (search)	Search-key aggregation	Not primary	Search	Emergency-only niche; ERDSAKCA generalizes revocation beyond emergencies

The game advances through the following levels [15]:

1. The adversary A starts by choosing a subset of the ciphertext classes S ⊆ {1, 2, ..., n} that it intends to attack. This set is fixed before seeing any public parameters, aligning with a selective security model.
2. The challenger B, in response, initiates the Setup-i algorithm for each i ∈ S, which involves generating the public parameters and public key PK_i, along with access control parameter U. These values are revealed to the adversary, while the corresponding msk remains confidential.
3. For each i ∈ S, the challenger also generates and provides the adversary with the aggregate decryption key K_S for all ciphertext classes not in S**,** ensuring the adversary has legitimate access to non-targeted classes.
4. In the Challenge-i phase, the challenger encrypts one of two challenge messages chosen by the adversary for each class **i ∈ S**, and the adversary must guess which message was encrypted.

In this security game, the adversary A begins by selecting a subset S ⊆ {1, 2, ..., n} of ciphertext classes it intends to attack. This choice is made before A observes any system parameters or public keys, ensuring a selective security model. The challenger B then executes the Setup algorithm to generate the system parameters, public key PK, and access control parameter U**,** which are provided to A. Additionally, B generates and gives A the aggregate decryption key K_S for all ciphertext classes not in S**,** granting legitimate access to those. For each i ∈ S**,** B encrypts a randomly chosen message based on a challenge bit, and A attempts to guess the bit. The adversary succeeds only if it correctly distinguishes all challenge ciphertexts in S. The scheme is semantically secure if this advantage is negligible over random guessing.

In our proposed approach, Generalized DHDHE [16] assumption is used to prove security because it better aligns with the needs of dynamic cloud environments. The original DHDHE assumption is designed for static settings with limited ciphertext-user mappings, making it less suitable for systems with frequent changes in user access and data classes. In contrast, the generalized DHDHE supports polynomial-size user and ciphertext class selections, making it more appropriate for dynamic, multi-user data sharing scenarios like ours. It captures the hardness of distinguishing challenge ciphertexts under complex and evolving key distributions, thereby enabling our ERDSAKCA scheme to achieve secure and scalable user revocation and access control. This broader assumption provides stronger semantic security guarantees against adaptive adversaries and ensures a robust security foundation as the system scales in complexity.

The first assumption is based on the Generalised DHDHE, where during the setup phase, let the parameters be 2n**.** Set $a_\ell = (2n+1) \cdot e_\ell$ for $\ell = 0, 1, ..., n-1$, and set $a_n = g^{\alpha(2n+1)} e_n$ for $\ell = n$, using $\alpha \in \mathbb{Z}_p$ as a random variable. Choose b_1 from $\mathbb{Z}_p$ and t_2 from $\mathbb{Z}_p$ at random, with $b = b_1 + t_2$, and set $a_1 = bt_1 n$ and $a_2 = bt_2 n$. Identifying $K = (2n)^{2n}$ from a random element in G_{2n} is the objective, given $\langle \forall i \in \{0, 1, ..., n\}, Y_1, Y_2, K \rangle$. For the generalized DHDHE problem, the advantages of a polynomial-time adversary A are described as:

$$\mathrm{Adv}_{\mathrm{A}}^{\mathrm{DHDHE}} = \left| \Pr\left[\mathrm{A}\left(\{X_i\}_{i\in\{0,1\ldots,n\}},\ Y_1, Y_2,\ K = g_{2n}^{at^{2^n}} \right) = 1 \right] - \Pr\left[\mathrm{A}\left(\{X_i\}_{i\in\{0,1\ldots,n\}},\ Y_1, Y_2,\ K = g_{2n}^{r} \right) = 1 \right] \right| \quad (1)$$

The above equation 1 describes the dynamic key generation for multi-users with polynomial user selection. This new assumption is clearly an extension of the DHDHE assumption, as can be seen from this. The specific reduction of the generalized DHDHE assumption to the DHDHE assumption can be achieved by multiplying a_1 and Y_2.

3.1.3 Notation Summary

To improve clarity and support understanding of the equations in the paper, Table 2 summarizes the key symbols and terms used throughout the mathematical formulations.

Table 2. Summary of Symbols and Notations

Symbol	Description
P, Q	Generators of bilinear groups G_1, G_2 respectively.
G_1, G_2, G_T	Cyclic groups of prime order p; G_T is the pairing result group.
e(·,·)	Bilinear pairing function $e : G_1 \times G_2 \rightarrow G_T$.
α, t, r	Random scalars from $\mathbb{Z}_p$.
A, Â	Number of logical data/user blocks.
B	Block size for data classes and users.
n = A × B	Total number of data classes.
m = Â × B	Total number of users.
PK, msk	Public key and master secret key.
Bsk	Broadcast secret key.
PK_a^i	Public key components for attributes.
$\gamma_a{}^1$, $\gamma_a{}^2$	Secret exponents
C_i	Ciphertext tuple (C_0, C_1, C_2, C_3).
S, S*	Sets of data classes and authorized user indices.
K_S	Aggregate key for data class subset *S*.
K(S, S*)	Broadcast key for users *S*.
$d_i{}^r$	User's private key.
aS_a, bS_a	Summations used in key derivation.
X_i, Y_1, Y_2	Public values in the DHDHE game.
$\mathrm{Adv}_{\mathrm{A}}^{\mathrm{DHDHE}}$	Adversary's advantage in distinguishing ciphertexts.

3.2 Process of Dynamic & Revocable Collaborative Key Sharing

This section describes an approach to dynamic key aggregate storage presuming n ciphertext classes with multiuser data sharing. While the size of the public parameters grows linearly with the number of ciphertext classes, our scheme guarantees that the ciphertext and aggregate key remain constant in size. Users' access rights can be dynamically revoked using the suggested scheme, all without requiring extensive changes to the system parameters. Along with the proposal, a security proof of concept is also offered. The use of the user revocation mechanism in KAC requires revocable key-aggregate encryption, as outsourced users in the cloud exhibit the characteristic of perpetual change.

When a user's credentials expire, they can have their access revoked using the Aggregate Dynamic & Revocable Key Cryptosystem (ADRKC), an improvement on key aggregate cryptosystems. Setup, KeyGen, Encrypt, UserRevoc, Update, Decrypt, and Verify are the seven polynomial-time algorithms that make up the dynamic and revocable key-aggregate encryption scheme [16, 17]. Their definitions are as follows:

1. The first input to the Setup algorithm is the maximum quantity of files n and the protection factor 1λ. It returns Params, the public parameter.
2. A Public Key (PK) and a master secret key (msk) are produced by a key generation algorithm by taking Params as input.
3. Take the PK, an index i representing the file, a message m, and the Params as inputs to the encryption algorithm. It generates a ciphertext C.
4. The UserRevoc algorithm accepts the msk, a set S of indices that correspond to various files, the user's identity uid, and the Params as input. Users' private key SK is outputted.
5. The update algorithm accepts the following parameters: a ciphertext C, a user revocation set R, and the Params. The result is a modified ciphertext a.
6. The following parameters are passed to the decryption algorithm: the cipher text C, the user private key SK, the set S, an index i representing the cipher text C, the user revocation set R, and the Params. Either the result u of 1 is output if ($i \in S$) $\wedge$ ($uid \notin R$).
7. The Verify algorithm accepts a cipher text C, an updated cipher text a, a PK, and Params as the input. If the revocation has been executed honestly and the cipher text has been updated correctly, the cloud server will output 1. Otherwise, it will output 0.

Figure 1 depicts the model developed for use with the ADRKC in a cloud setting. There are three parts to it: the user, the owner, and the cloud service provider.

In order to obtain the system parameters, data owner Alice runs the setup algorithm. Then, using a revocable key aggregate encryption system, several files u1, u2, .. can be shared with other people over the cloud server.

After that, Alice secretly stored msk and used KeyGen (params) to generate a random public/master secret key-pair (PK, msk).

Afterward, by running Encrypt (PK, i, R, m, params) on the cloud server, Alice and anyone else who helped Alice can upload the encrypted files. Once Alice decides to share multiple files with Bob, Alice will run the UserRevoc algorithm (msk, uid, S, params) to generate SK for Bob based on the indices of authorized files and the user's identity. Due to the fixed size of SK, Alice can easily send it to Bob through a secure channel at a low communication cost. The user revocation list R will be sent to CSP by Alice whenever she wishes to remove users. In order to update the corresponding ciphertext, CSP then invokes the algorithm Update (PK, R, a, and params). Bob's revoked access must not be disabled before downloading the new ciphertext from the cloud server and using the private key to run the Decrypt algorithm (O, SK, T, i, R, params) to extract the actual text. If the user's access has been cancelled, like David's in Figure 1, he will lose access to the files because he cannot decipher the updated ciphertext. At last, by using the algorithm Verify (a, O, params), Alice can confirm that the user revocation is successfully completed and check the changed ciphertext. Such verifiability in outsourced settings is similarly emphasized in [18], though the model lacks integration with dynamic ciphertext updates.

Dynamic access control: The ability to dynamically update user access to a set of ciphertexts is a key component of the proposed scheme. To remove a user's access to a set of ciphertext classes after they have been granted an aggregate key by the data owner in KAC, the owner must first change the master secret key. However, it is expensive and perhaps problematic to change the msk each time a user's ciphertext class access rights need to be updated. Our plan, however, addresses this issue through enabling the owner of data to instantly modify permissions of user.

To accomplish this, in the proposed scheme, the parameter U = tP is not included in the ciphertext but rather in the aggregate key. Any encrypted ciphertext class in subset S can be decrypted by the user if they have the correct value of U. Imagine for a moment that the data owner wants to change who can access subset S.

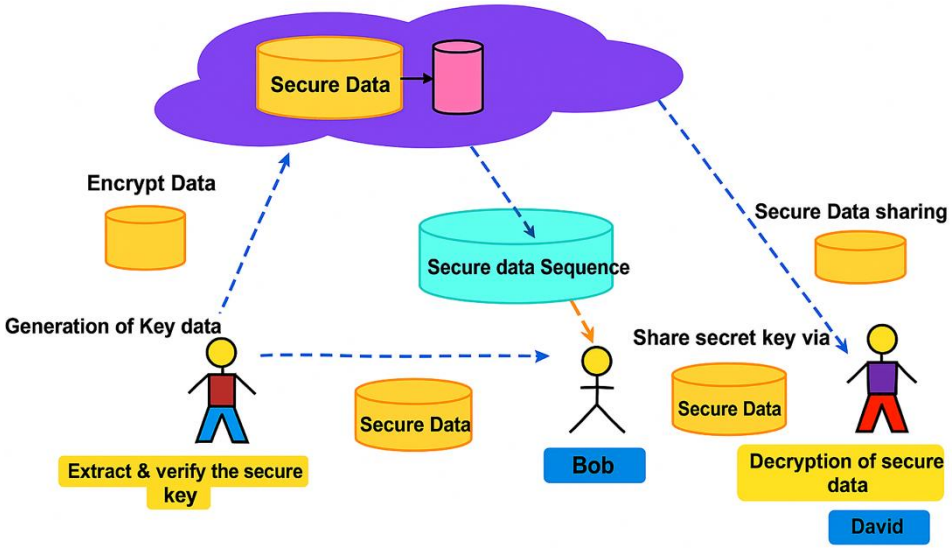

Figure 1. General Processes of Dynamic and Revocable Multi Data Sharing with Aggregate Key Cryptosystem.

She has the option to reencrypt all the ciphertexts in that class with a new random element t′ ∈ Zq. Then, she can only grant access to the users she wants by providing them with the updated dynamic access parameter U′ = t′P. If you use the same t for encryption and decryption, the decoded value only will provide the exact information m. In contrast, some earlier techniques could not control access privileges effectively, as they relied solely on the random parameter embedded in the ciphertext. This represents a significant difference from the proposed method. Moreover, the parameter U remains constant in size and is transmitted only when updated, ensuring that system performance remains largely unaffected.

3.3 Construction of Efficient Revocable & Dynamic Secure Aggregate Key Cryptosystem Approach (ERDSAKCA)

This section describes the construction of the proposed Efficient, Revocable, and Dynamic Secure Aggregate Key-based Cryptosystem for Access control (ERDSAKCA) encryption scheme, designed for secure, efficient, and revocable multiuser file sharing in cloud environments. The scheme outlines the implemented processes, security level performance, multiuser revocation, and dynamic access control in data sharing.

3.3.1 Process

The implemented process as follows:

1) Setup of Cloud Environment: Pick α ∈ Zq at random using SetUpB (1^{λ}, n, m). The system parameter should be output shown in Algorithm 1 as The parameters are set to include the following: P, Q, Y P, α, B, Y Q, α,BC. Throw out α. Find A as ⌊n/B⌋ and A′ as ⌊m/B⌋.

Algorithm 1 Algorithm process of cloud setup environment

1) Create a bilinear map group system B =
2) (p, G, GT, e (· , ·)).
3) Create a set of public parameters called PubK with the following definition: g = gol ∈ G for every l ∈ {1, 2, . . . , n, n + 2, . . . , 2n}.
4) Choose a collision-resistant hash function H1: {0, 1} × {0, 1} → {0, 1}** that is one-way.
5) H2: {0, 1} × {0, 1}** → is a collision-resistant hash function that must be monotone minimum, and set SP ←
6) (B, PubK, H1, H2).
7) KS ← γ {0, 1} | Γ = 0.

2) Gen of User Key: Probably select the attributes: γ1, . . . , γA ∈ Zq, & γ1′ , . . . , γA′ ∈ Zq. msk1 = (γ1, . . . , γA) & msk2 = (γ1′ , . . . , γA′). After that, specify for 1 ≤ a ≤ A and 1 ≤ a′ ≤ A, as shown in equation 2 & 3.

$$PK_1^a = \gamma_1^a P. PK_2^a = \gamma_1^a Q$$
$$PK_3^a = \gamma_2^a P. PK_4^a = \gamma_2^a Q \quad (2)$$
$$PK_1 = (PK_1^1, \dots, PK_1^A)$$
$$PK_2 = (PK_2^1, \dots, PK_2^A)$$
$$PK_3 = (PK_3^1, \dots, PK_3^A) \quad (3)$$

As msk = (msk1, msk2), print the master secret key, and as PK = (PK1, PK2, PK3, PK4), print the public key. Uniformly at random choose the secret broadcast key bsk = γ3 from Zq and provide it as additional output.

3) Encrypt of Owner Data: Parameterized function t (PK, i, M): find a = ⌊i/B⌋ and b is calculated as (i mod B) + 1. Consider a-th element of PK_2 as PK_{a2}.

Algorithm 2 Encryption of Files for Secure Data Sharing in the Cloud

1) Choose a value t from the set Zp.
2) If the value l is from the set {1, 2, . . . , }, then
3) C11 ← gt.
4) C12 ← (v. g) the fifth time: δl = (C11, C12).
5) C1KW1 ← ϕwi, jϵ ← for all integers from 1 to x in 6.
6) For every i and j in the set ← KW, do the following:
7) C1KWi ← H2(wi, j) · e (gl, gn). Eleventh: Cl ← The equation KW = C13 ← (C1KW1, {C1KWi}) where i ∈ {x + 1, x + 2, . . . , y} and wi, j ∈← KW
8) e ← KW
9) C1 = (C11, C12, C13, C14)
10) The owner of the data keeps the ciphertext C1 on the server in the cloud.

Select an element t from the set Zq at random and print out the partial cipher text C0 as shown in equation 4.

$$C = (C_0, C_1, C_2, C_3) = \left(tQ, tPK_2^a, t(PK_2^a + Q), M \cdot e(P_b, tQ)\right) \quad (4)$$

Conversion of cipher text from random user attributes using the above equation.

Enter the partially encrypted cipher text C0 = (c_0, c_1, c_2, c_3), the msk = (msk1, msk2), and the bsk as arguments to the System Encrypt function shown in algorithm 2.

Here, the class id i is the extra piece of data that is needed.

With a = ⌊i/B⌋, we can denote the a-th component of msk1 as $msk1^a$.

Show the decipher text C′ in its final form as specified in equation 5.

$$C' = (C_0', C_1', C_2', C_3')$$
$$= (c_0, c_1' - (bsk \cdot msk1^a)Q, c_2, c_3)$$
$$= (tQ, (t - bsk)PK_2^a, t(PK_2^a + Q_b), M.e(P_b, tQ_1)) \quad (5)$$

This is formation of decrypted text with cipher text from equation 4.

4) Password Generator (param, msk, ˆi): The user's private key, denoted as d′, should be displayed as shown in equation 6.

$$d_i' = msk2_b^{a'}, \; P = \gamma_{2b}^{a'}, \quad (6)$$

This is the same setting d′ to a′BPKA 3 in an indirect way.

5) User Revoc of Key (param, msk, S): Suppose msk is equal to (msk1, . . . , mskA). The set Sa is defined as {i mod B + 1|i ∈ S, di/Be = a} for the part of indices of the class S where 1 ≤ a ≤ A.

Next, calculate for all values of a from 1 to A, evaluate the values from revocable keys for users as shown in equation 7.

$$K_S^a = msk_1^a \sum_{j \in S_a} P_{B+1-j} = \gamma_1^a \sum_{j \in S_a} P_{B+1-j} \quad (7)$$

Output of the final attributes sequences as follows:

$$K_S = (K_S^1, \ldots, K_S^A) \quad (8)$$

6) Multi-User Broadcast Secure Data Share (param, KS, S*, PK, bsk): The following is a broadcast to all users in S* of the aggregate key KS = (K1 S, …..KAS). Define S* as the set of user ids where 1 ≤ a* ≤ A* and for each i in S*. define a∗ as the product of {i mod B + 1|i ∈ S, di/Be = a}.

Pick t∗ at random from the set Gq and set it for 1 ≤ a∗ ≤A∗.

$$b_{s_a} = \sum_{j \in S_a} Q_{B+1-j} \quad (9)$$

Based on the above sequences (equation 9), generated output collaborative key as shown in equations 10 & 11.

$$K_{(S,S^*)} = (tQ, K_1, K_2) \quad (10)$$

Here

$$K_1 = \left(t\left(PK_1^4 + b_{S_1}\right), \ldots, t\left(PK_1^4 + b_{S_A}\right)\right)$$
$$K_2 = (\{e(P_B, tQ_1) \cdot e(K_s^a, Q)^{bsk}\}_{1 \le a \le A}) \quad (11)$$

Note that K (S, S*) now comprises of O (A + A*) group elements.

Algorithm 3. Decryption of Files for Secure Data Sharing in the Cloud

1) If i is a member of the set S, then do the following:
2) If pub ∈ S is true, then
3) ŷ ← Sig [1 ‖ i] [eSig′].
4) R1 ← e(Tr_1 · pub^{-1}, C1)/e(pub^{-2}, C2).
5) e(C1KWI, Tr2) with i × x + y and y′.
6) Hii ← C1KWI [e(C4, Tr2)e(pub^{-2}, C12)e(pub^{-4}, C11)] where Tr0, pub^{-1}, and pub^{-3} are the inputs and C1 is the output.
7) The set pub i ∈ Sig [n + 1 − j] is defined as ŷ ∈ S, where j = lg [n + 1 − j].
8) Ri ← 1 if (Hi > Tri1) and (Hi < Tri2).
9) Include Hii in the Hash Value List (HVL):
10) Ri ← 0 "Ri equals zero."
11) For every i in the set f × 1 × 2…. y′, execute the following:

12) If RiA = 1, then add i to the Search Result List (SRL).
13) The return failure case.

7) Decrypt of Keys: The function t (Param, C, K(S, S′), i′, di′, S, S′) is defined as follows: The output will be zero if either i′ is an element of S or i′ is an element of S′. If not, then multiply di′/Be by i′ mod B + 1 in algorithm 3, and then set:

$$aS_a = \sum_{j \in S_a \setminus \gamma_b} P_{b+1-j+b}, \quad (12)$$

$$bS_a = \sum_{j \in S_a} P_{b+1-j}, \quad (13)$$

The set C can be defined as (c0, c1, c2, c3) and

$$K_{\{(S,S')\}} = \left(k_{\{0\}}, \left(k^{\{1\}}, \ldots, k^{\{4\}} \right), \left(k^{\{1\}}, \ldots, k^{\{4\}} \right)_{\{2\}} \right) \quad (14)$$

Back to the decrypted message as follows:

$$M = C_3 K_2 . \frac{e(b_{S_a} c_1)\, e(a_{S_a} c_0)}{e(b_{S_a} c_2)} . \frac{e(d_i + a_{S_a} k_0)}{e(P_{b+1} k^1)} \quad (15)$$

The framework mentioned earlier is designed for use by a sole owner of the data. To accommodate m0 owners of data, m0 such setups must be initiated similar to the multi-owner KASE approach described in [19]. Proof of validity for this construction can be skipped, as it mirrors the construction of fundamental KAC.

The following theorem addresses the non-adaptive CPA security of the generalized extended KAC construction. Let G_1 and G_2 be two subgroups of order q of a bilinear elliptic curve. The broadened and enhanced KAC, which handles n data classes and m users, is (τ, B)-CPA secure if the non-symmetric decision (τ, B, B)-BDHE assumption holds in (G_1, G_2) for any positive integer triple (n, m, B) where B ≤ min (n, m).

Additionally, the extended KAC construction can be further developed to achieve Chosen Ciphertext Attack (CCA) security using appropriate techniques.

3.3.2 Security Level Performance

An essential part of the system's performance is the selection of A, A0, and B. The construction makes it very evident that there is a constant number of group elements that make up the ciphertext. There are O(B) group elements in the public parameter and O(A + A′) in the public key PK and the broadcast aggregate key K(S, S′). So, for uses that necessitate minimal overhead aggregate keys, a smaller value of B is beneficial. The ordering notation for the storage complexities of generalized methods is summarized in Table 3. The space complexity of any group member in G1, G2, and GT is assumed to be O(n1), O(n2), and O(nT), respectively.

Table 3. Complexity of Space with Processing of Key Aggregate

Parameter	Complexity of the Sequence
Param	$O(B(n_1 + n_2))$
msk	$O(A + A_1 \log q)$
PK	$O(A_{n1}+A_{n2})$
bsk	O(log q)
c	$O(n_1+n_2)$
$K_{(S,S')}$	$O((A+1)n_2+An_1)$

3.3.3 Multi User Revocation & Dynamic Access in Data Sharing

The foundation of our structure is a component that allows B data classes and users to function. This allows the system to manage n = A×B data classes and m = Â×B data users. The system is designed to plan and execute A × Â instances of this component simultaneously.

Although each building block uses its own private and public key components, they all share the same set of public parameters. The construction maintains constant ciphertext overhead while balancing the size of the public parameters with the aggregate key and public key sizes.

Adding and removing users from an existing list is especially important in multi-user contexts, where both the total number of users and their individual permissions to access resources are subject to ongoing change [20], including emergency access scenarios using break-the-glass delegation. The extended KAC construction seamlessly handles the addition of new users as shown in the Figure 2. To add a new user, simply generate an updated key for each newly joined user. From that point onward, all broadcast operations of the aggregate key will account for the access rights of both recent and existing users.

Notably, the system does not need to recreate a basic size-B block in the enlarged framework to accommodate new users. Adding new users only increases the value of parameter A, which is equivalent to creating additional instances of the same basic building block. Consequently, there is no need to modify the existing owner/user keys or the public parameters. This compatibility with broadcast encryption ensures the system can handle an increased number of users without complications.

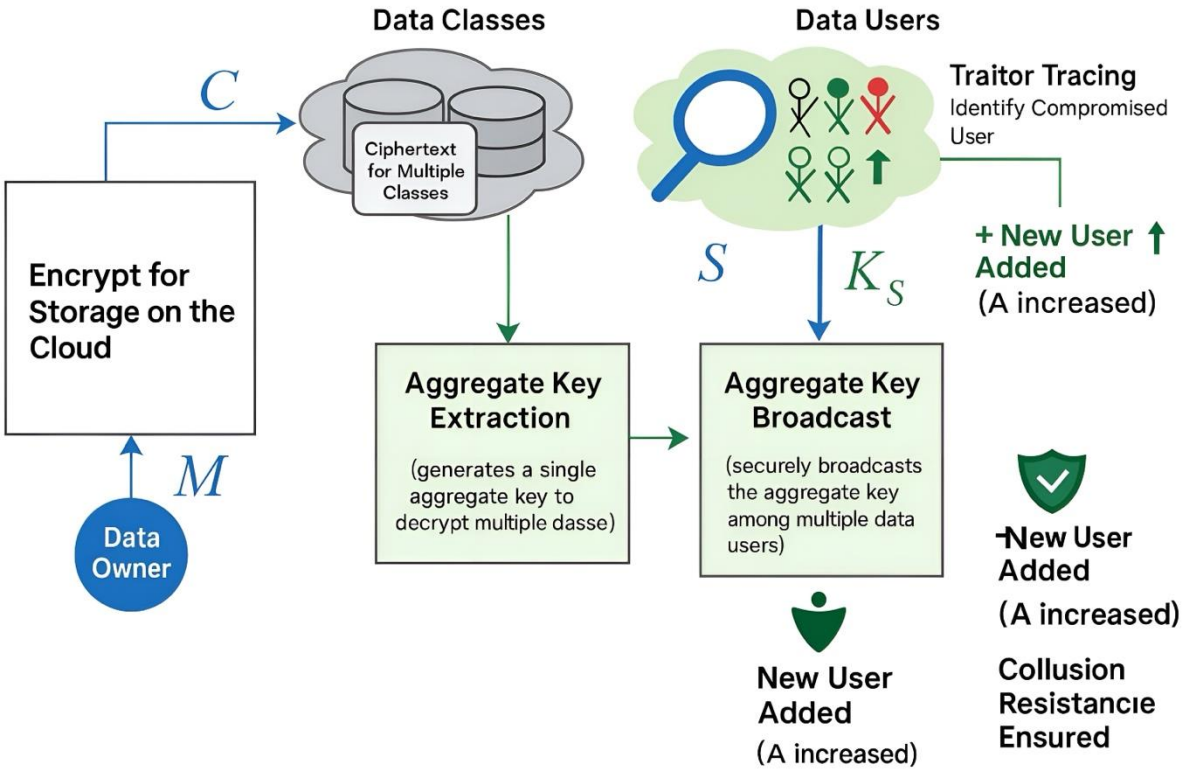

Figure 2. Extended KAC with User Management, Revocation and Traitor Tracing Features

User revocation, a fundamental feature of broadcast encryption systems, is also achieved at no additional cost. For instance, if a data owner decides that a specific user should no longer access any documents uploaded to the cloud, they can instruct the system to exclude that user's identity from future broadcast operations. This effectively prevents the revoked user from accessing aggregate keys associated with any new data the owner uploads to the cloud. The corresponding security guarantee comes naturally from the collision resistance property of the extended KAC construction, which is not incurable at all. Specifically, the collusion resistance property ensures that a revoked user's prior knowledge of aggregate Keys do not compromise access to future documents, as their access rights do not cover the new indexes.

While revoked users naturally retain access to previously existing documents (which they could have downloaded and saved prior to revocation), revoking access to such documents is often unnecessary. However, if the data owner wishes to update an existing document and prevent a revoked user from accessing it, they can assign a different index to the updated document.

Revocation is commonly employed when rogue or compromised users threaten the system's security. Tracing such users, however, is more complex and requires traitor-tracing systems. Consider a scenario as shown in Figure 2 in which the data owner shares an aggregate key for a specific subset S of plaintext data with a group of authorized users Ŝ. A potential threat arises if a malicious actor compromises one of the users in the receiver set Ŝ, obtains their secret key, and develops a decoder that unauthorized parties can use to access the plaintext.

To address this, the data owner can employ a tracing technique that uses a publicly available malicious decryptor to identify at least one compromised user index and revoke their access. A traitor-tracing system is valuable in this context. Although this study does not aim to provide a comprehensive description of the integration of the extended KAC framework with a traitor tracking scheme, it should be noted that the extended KAC framework could potentially leverage several proposals from the literature that have explored integrating traitor tracking with broadcast encryption, with varying degrees of success.

4. Results

The performance of the proposed ERDSAKCA scheme is evaluated against two closely related and widely cited schemes, Key-Aggregation Authorized Searchable Encryption Scheme (KAASE) [11] and Multi-Owner Key-Aggregate Searchable Encryption Scheme (MOKASE) [19], which represent the state-of-the-art in key-aggregate searchable encryption with revocation and multi-user capabilities. The evaluation focuses on key metrics, including setup time, key generation, encryption, decryption, user revocation, trapdoor generation, and memory utilization, as illustrated in Figures 3 through 8.

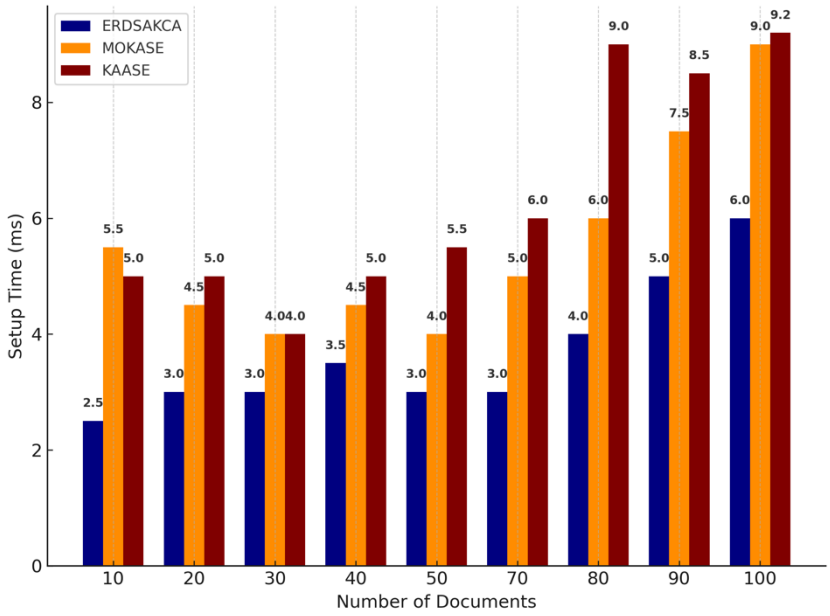

Figure 3. Performance evaluation of cloud setup environment

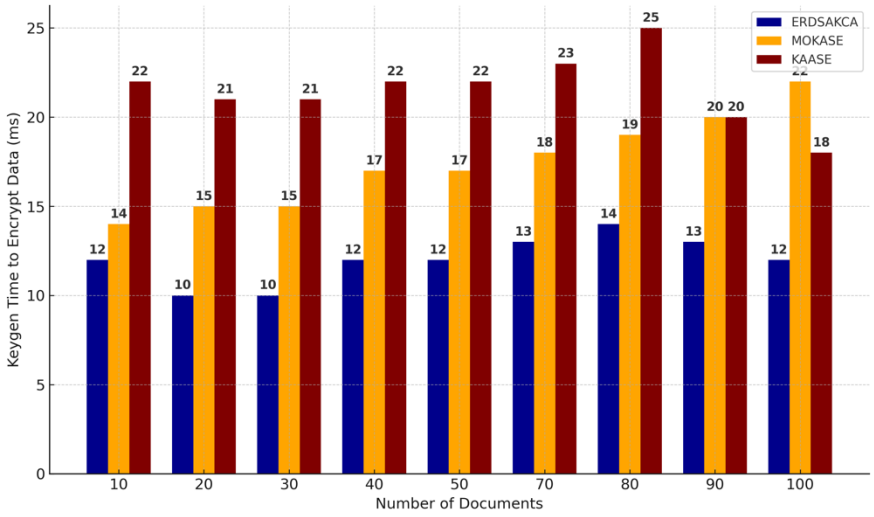

Figure 4. Key generation time based on different file attributes

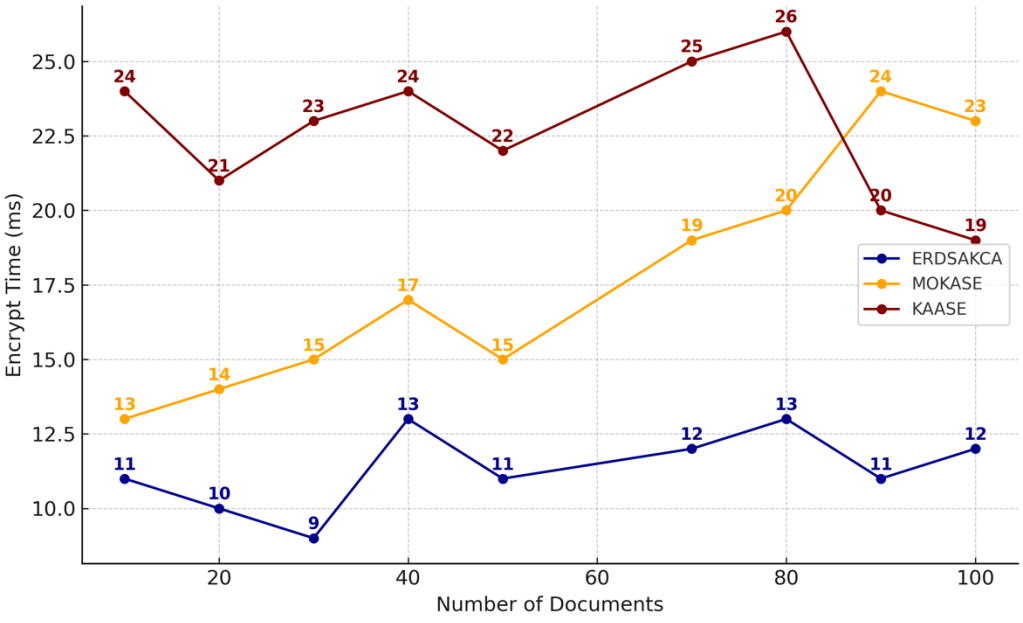

Figure 5. Encryption time to explore files to cloud

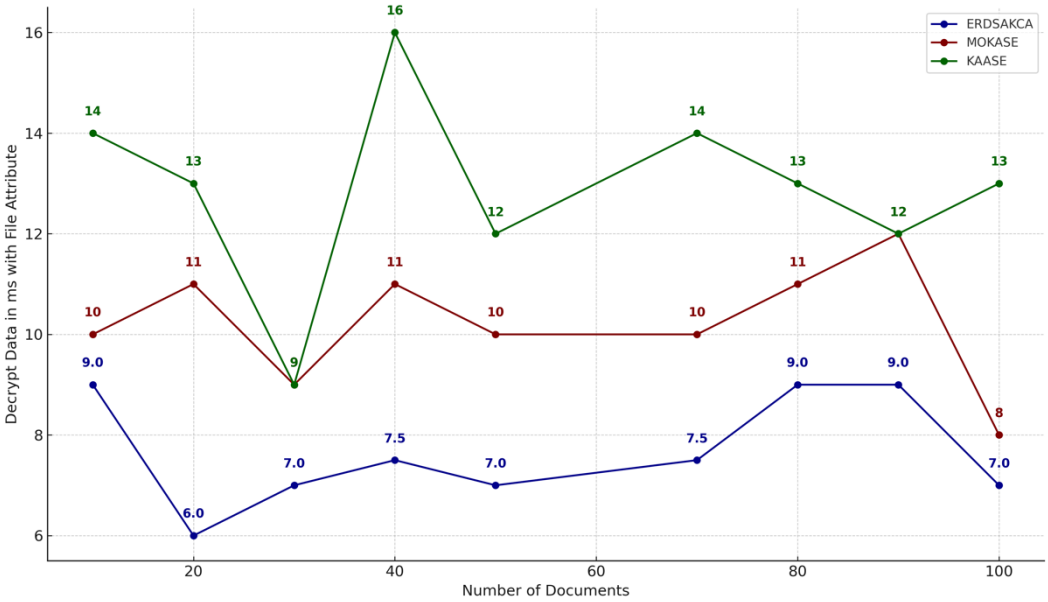

Figure 6. Performance of data extraction / decrypted data with different files attributes

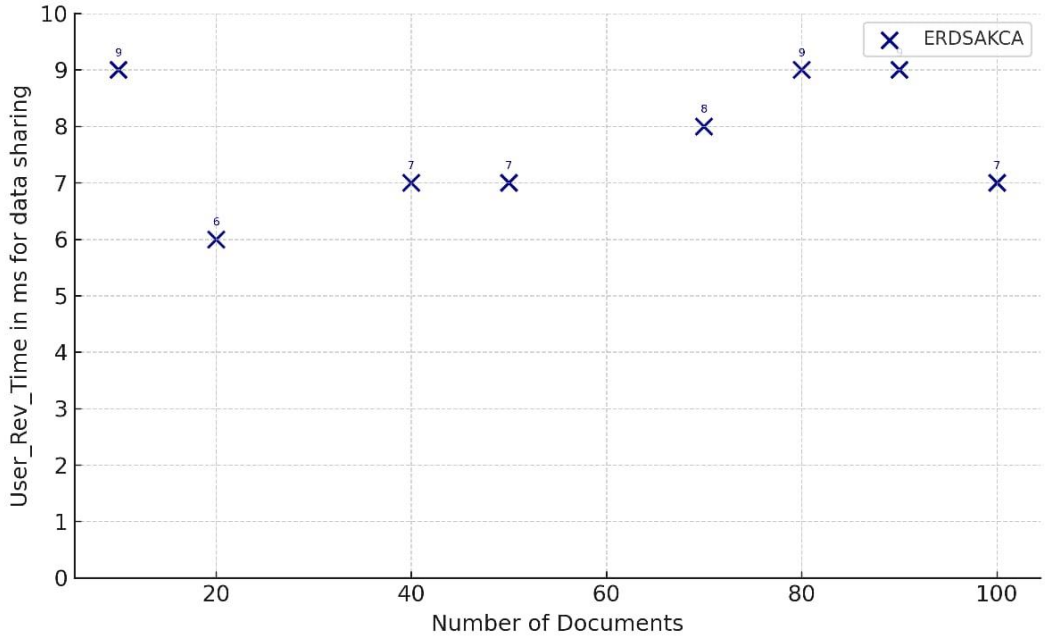

Figure 7. Performance of proposed approach in term of user revocation

For the cloud environment setup, the implementation was carried out using the latest version of CloudSim, with Java and NetBeans as the development tools. Experiments were conducted with document counts ranging from 10 to 100 in increments of 10, each document limited to 10 KB and containing randomly selected words from a dictionary.

Cloud Setup Environment: Figure 3 shows the cloud setup time for ERDSAKCA, MOKASE, and KAASE as the number of documents increases. ERDSAKCA consistently exhibits the lowest setup time, ranging from approximately 3 ms to 6 ms, compared to MOKASE (4 ms to 9 ms) and KAASE (4 ms to 9 ms). The setup time for ERDSAKCA grows linearly with the number of documents but remains significantly lower, demonstrating its efficiency in initializing the cloud environment.

Key Generation Based on Different Attributes: As depicted in Figure 4, the key generation time for ERDSAKCA remains nearly constant, fluctuating between 10 ms and 14 ms across varying document counts. In contrast, MOKASE and KAASE show slightly higher times, ranging from 14 ms to 20 ms and 18 ms to 25 ms, respectively. This near-constant performance of ERDSAKCA highlights its scalability in generating keys, even as the number of attributes increases.

Encryption Time to Explore Files in the Cloud: Figure 5 illustrates the encryption time for ERDSAKCA, MOKASE, and KAASE.

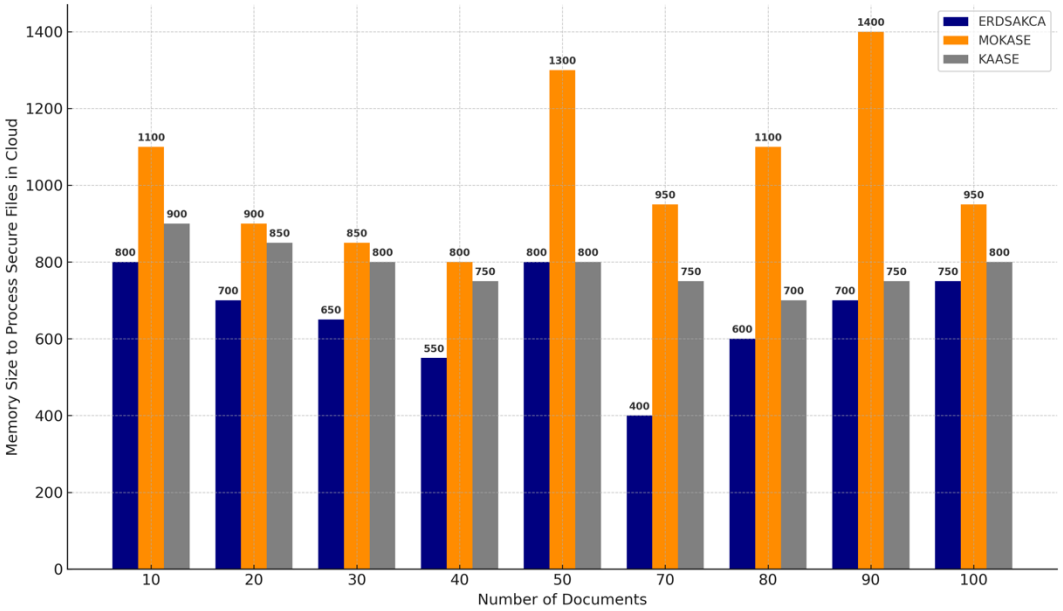

Figure 8. Performance of data size memory utilization for processing secure data in cloud

Table 4. Comparative Summary of ERDSAKCA with MOKASE and KAASE

Feature	MOKASE	KAASE	ERDSAKCA (Proposed)
Key Management	Single key per owner	Attribute-based, fixed-size	Constant-size key; aggregate key broadcast
Dynamic User Revocation	Limited, no ciphertext update	Moderate, requires partial updates	Efficient, revokes via ciphertext update
Verification Mechanism	Not integrated	Basic hash-based	Integrated ciphertext update verification
Ciphertext Size	Variable	Constant	Constant
Trapdoor Generation	Supported	Supported	Supported
Semantic Security	BDHE-based	ABE with trapdoor protection	Generalized DHDHE-based
Multi-User Multi-File Support	Yes	Limited	Fully supported
Scalable Cloud Deployment	Partial	Limited	Fully scalable and lightweight
Setup Time	4–9 ms	4–9 ms	3–6 ms
Key Generation Time	14-20 ms	18-25 ms	10-14 ms
Encryption Time	13–24 ms	19–26 ms	9-13 ms
Decryption Time	8-12 ms	9-16 ms	6-9 ms
Memory Usage	800-1400 KB	750-900 KB	400-800KB
User Revocation Time	High	Moderate	Low (6–9 ms)

ERDSAKCA maintains a lower encryption time, averaging between 9 ms and 13 ms, compared to MOKASE (13 ms to 24 ms) and KAASE (19 ms to 26 ms).

The linear increase in encryption time with the number of documents is evident across all schemes, but ERDSAKCA's use of a single pairing operation results in a reduced computational overhead compared to the pairing-heavy approaches of MOKASE and KAASE.

Performance of Data Extraction/Decryption with Different File Attributes: Figure 6 illustrates the decryption performance of three schemes, ERDSAKCA, MOKASE, and KAASE across varying numbers of documents. ERDSAKCA consistently outperforms the other two, maintaining the lowest and most stable decryption times between 6-9 ms. MOKASE shows moderate performance with decryption times ranging from 8-12 ms, while KAASE exhibits the highest and most fluctuating times between 9-16 ms. Overall, ERDSAKCA demonstrates superior efficiency and scalability, making it the most effective scheme for handling increasing document volumes.

The graph in Figure 7 illustrates the user revocation time (in milliseconds) for the ERDSAKCA scheme across varying numbers of documents. The revocation time fluctuates slightly but generally remains within a range of 6 to 9 ms, showing no clear upward or downward trend as the number of documents increases. This indicates that the proposed approach maintains a consistently low and stable revocation time, even as the data size scales, which demonstrates its efficiency and reliability in managing user revocation during data sharing.

Performance of Data Size Memory Utilization for Processing Secure Data in the Cloud: Figure 8 compares memory utilization. ERDSAKCA uses less memory compared to MOKASE and KAASE.

To illustrate the practicality and advantages of ERDSAKCA, a comparative summary is presented in Table 4, highlighting key findings such as cloud setup time, encryption time, decryption time, user revocation time, and memory usage, along with core security, efficiency, and system-level features relative to two existing schemes, MOKASE and KAASE. This direct comparison emphasizes the unique strengths of ERDSAKCA in enabling multi-user access control, scalable deployment, and low computational overhead.

5. Discussion

The comparative analysis presented in Table 4 clearly demonstrates that ERDSAKCA outperforms both MOKASE and KAASE in terms of efficiency, especially in dynamic environments with multi-user and multi-file access control requirements.

First, ERDSAKCA offers robust dynamic revocation by updating ciphertext alone, eliminating the need for costly re-keying for authorized users which is an advantage not offered in either KAASE or MOKASE. Its constant-size aggregate key simplifies key management complexity while reducing transmission and storage overhead.

Second, ERDSAKCA's encryption and decryption times remain consistently low across all file sizes. Unlike KAASE, which depends heavily on pairing operations, ERDSAKCA minimizes computational load with a single pairing operation per file, significantly improving processing efficiency.

Third, the built-in verification mechanism ensures that ciphertext updates and revocation operations are executed honestly, which is an essential security enhancement absent in earlier models.

In terms of scalability, ERDSAKCA's lightweight design allows smooth integration with large-scale cloud systems. Its lower memory consumption relative to MOKASE and KAASE makes it highly suitable for resource constrained environments.

The performance advantages observed in ERDSAKCA, including reduced encryption and decryption times, efficient user revocation through ciphertext updates, and lower memory usage, align with recent research developments in revocable and aggregate key cryptographic systems [5, 6]. However, ERDSAKCA distinguishes itself by integrating a built-in ciphertext verification mechanism, addressing notable security gaps highlighted in recent studies [7, 12]. Furthermore, unlike existing models such as [11, 19], ERDSAKCA consistently maintains stable key-generation and setup times across increasing document sizes, underscoring its practicality and scalability in dynamic multi-user cloud environments.

6. Conclusion

This article proposes the Efficient Revocable Dynamic Secure Aggregate Key Cryptosystem Approach (ERDSAKCA), which effectively implements key aggregation and user access control in a cloud environment based on a dynamic and revocable key-aggregate cryptosystem. The proposed method updates the ciphertext on cloud servers, enabling revocation of user permissions without requiring legitimate users to change their secret keys. In addition, an extended system is introduced to accommodate the cloud environment, where the number of files is large and grows rapidly. A verification mechanism ensures that user revocation is executed correctly. Furthermore, the fundamental Key-Aggregate Cryptosystem (KAC) architecture is shown to be easily generalizable and expandable, allowing secure broadcasting of the aggregate key to multiple users in a real-world data-sharing scenario. This lays the foundation for developing a flexible and scalable public-key based system for online data exchange in the cloud. According to the performance evaluation, the proposed approach achieves efficient user access control while significantly reducing transmission and storage costs compared to existing schemes. An additional enhancement of the proposed approach is the provision of identity-based privacy, addressing multi-user data conflicts in cloud computing data sharing.

References

[1] A.Q. Khan, M. Matskin, R. Prodan, C. Bussler, D. Roman, A. Soylu, Cloud storage cost: a taxonomy and survey. World Wide Web, 27(36), (2024) 1–33. https://doi.org/10.1007/s11280-024-01273-4

[2] P. Aryan, S.D. Shetty, Designing a secure, scalable, and cost-effective cloud storage solution: A novel approach to data management using NextCloud. TrueNAS, and QEMU/KVM, International Conference on Computational Intelligence and Network Systems (CINS), IEEE, United Arab Emirates. https://doi.org/10.1109/CINS63881.2024.10864401

[3] E. Alhelali, K.M. Ramokapane, J. Such, Multiuser privacy and security conflicts in the cloud. CHI '23: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, (2023) 1-6. https://doi.org/10.1145/3544548.3581307

[4] L. Golightly, P. Modesti, R. Garcia, V. Chang, Securing distributed systems: A survey on access control techniques for cloud, blockchain, IoT and SDN. Cyber Security and Applications, 1, (2023) 100015. https://doi.org/10.1016/j.csa.2023.100015

[5] K. Zhang, X. Hu, J. Zhao, L. Wei, J. Ning, Blockchain-based revocable key-aggregate searchable encryption for group data sharing in cloud-assisted Industrial IoT, IEEE Internet of Things Journal, 12(11), (2025) 16899–16911. https://doi.org/10.1109/JIOT.2025.3534837

[6] J. Liu, J. Qin, X. Zhang, H. Wang, Efficient key-aggregate cryptosystem with user revocation for selective group data sharing in cloud storage, IEEE Transactions on Knowledge and Data Engineering, IEEE, 36(11), (2024) 6042–6055. https://doi.org/10.1109/TKDE.2024.3397721

[7] G. Pareek, B.R. Purushothama, KAPRE: Key-aggregate proxy re-encryption for secure and flexible data sharing in cloud storage. Journal of Information Security and Applications, 63, (2021) 103009. https://doi.org/10.1016/j.jisa.2021.103009

[8] J. Zhao, Q. Su, Verifiable data sharing scheme for dynamic multi-owner setting. arXiv preprint, arXiv:2308.00239 (2023) 113-125. https://doi.org/10.5121/csit.2023.131309

[9] M. Padhya, D.C. Jinwala, MULKASE: A novel approach for keyaggregate searchable encryption for multi-owner data. Frontiers of Information Technology & Electronic Engineering, 20(12), (2019) 1717–1748. https://doi.org/10.1631/FITEE.1800192

[10] M. Padhya, D.C. Jinwala, R-OO-KASE: Revocable online/offline key aggregate searchable encryption. Data Science and Engineering, 5(4), (2020) 391–418. https://doi.org/10.1007/s41019-020-00136-y

[11] H. Wang, KAASE: Key-aggregation authorized searchable encryption scheme for multi-key encryption data sharing, SSRN preprint (2022). https://dx.doi.org/10.2139/ssrn.4063519

[12] K. Alimohammadi, M. Bayat, H.H. Javadi, A secure key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage. Multimedia Tools and Applications, 79(3), (2020) 2855–2872. https://doi.org/10.1007/s11042-019-08292-8

[13] S. Yao, R.V. Dayot, H.J. Kim, I.H. Ra, A novel revocable and identity-based conditional proxy re-encryption scheme with ciphertext evolution for secure cloud data sharing. IEEE Access, 9(2021) 42801–42816. https://doi.org/10.1109/ACCESS.2021.3064863

[14] J. Liu, J. Qin, W. Wang, L. Mei, H. Wang, Key-aggregate based access control encryption for flexible cloud data sharing. Computer Standards & Interfaces, 88, (2024) 103800. https://doi.org/10.1016/j.csi.2023.103800

[15] M. Kamimura, N. Yanai, S. Okamura, J.P. Cruz, Key-aggregate searchable encryption revisited: formal foundations for cloud applications and their implementation. IEEE Access, 8, (2020) 24153–24169. https://doi.org/10.1109/ACCESS.2020.2967793

[16] Q. Gan, X. Wang, D. Wu, Revocable key-aggregate cryptosystem for data sharing in cloud. Security and Communication Networks, 2017, (2017) 1–11. https://doi.org/10.1155/2017/2508693

[17] S. Patranabis, Y. Shrivastava, D. Mukhopadhyay, Dynamic key-aggregate cryptosystem on elliptic curves for online data sharing. In International conference on cryptology in India, Springer International Publishing. https://doi.org/10.1007/978-3-319-26617-6_2

[18] X. Wang, X. Cheng, Y. Xie, Efficient verifiable key-aggregate keyword searchable encryption for data sharing in outsourcing storage. IEEE Access, 8, (2019) 11732-11742. https://doi.org/10.1109/ACCESS.2019.2961169

[19] T. Li, Z. Liu, C. Jia, Z. Fu, J. Li, Key-aggregate searchable encryption under multi-owner setting for group data sharing in the cloud, International Journal of Web and Grid Services, 14(1), (2018) 21–43. https://doi.org/10.1504/IJWGS.2018.088358

[20] M. Padhya, D.C. Jinwala, (2019) BTG-RKASE: Privacy preserving revocable key aggregate searchable encryption with fine-grained multi-delegation break-the-glass access control. Proceedings of the 16th International Joint Conference on e-Business and Telecommunications – SECRYPT, 2, 109–124. https://doi.org/10.5220/0007919901090124

Authors Contribution Statement

Sameera Mahammad: Conceptualization, Validation, Investigation, Writing - Original Draft, Visualization. K. Usha Rani: Supervision, Project administration, Validation, Writing - Review & Editing. Both the authors approved the final version of the work.

Funding

The authors declare that no funds, grants or any other support were received during the preparation of this manuscript.

Competing Interests

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Data Availability

The data supporting the findings of this study can be obtained from the corresponding author upon reasonable request.

Has this article screened for similarity?

Yes

About the License