DOI: 10.21917/ijsc.2026.0574

AN ENHANCED CARSA FRAMEWORK FOR SECURE AND EFFICIENT ACCESS TO OUTSOURCED CLOUD DATA

P. Abinaya[1], J. Senthil Kumar[2], P. Swathika[3] and J.A. Joe Antony Celshiya[4]
[1,4]Department of Computer Science and Engineering, Mepco Schlenk Engineering College, India
[2]Department of Electronics and Communication Engineering, Mepco Schlenk Engineering College, India
[3]Department of Artificial Intelligence and Data Science, Mepco Schlenk Engineering College, India

Abstract

Sensitive data exposure on the Internet has increased due to the quick development of Internet-based services and applications. Users can save money and save time by using cloud storage instead of keeping local storage, however there are significant privacy and security issues. Data owners may create fine-grained access controls with Ciphertext-Policy Attribute-Based Encryption (CP-ABE), which guarantees that only authorized users can decode the data. This makes it a potential solution. For further protection, especially in cloud contexts, CP-ABE can be used with RSA to add temporal data sensitivity, fine-grained access control, and robust cryptographic foundations. In addition to enforcing stringent access control, this study suggests a unique CP-ABE with RSA (CARSA) approach that identifies the users participating in the decryption process. The proposed CARSA system is shown to be more efficient by experimental assessment, with encryption and decryption timings of 1610 ms and 1098 ms for a 256-bit key length, respectively. By contrast, current methods take 1134 ms and 1691 ms, respectively. According to the findings, CARSA offers improved speed in cloud-based settings while guaranteeing strong data security.

Keywords:

Outsourced Data, Encryption, Decryption, RSA, Secure Data Access

1. INTRODUCTION

Cloud computing is commonly used in many industries to advance science and technology. Cloud computing provides strong tools for internet users and calculates a bigger scale of complicated data [1]. Data is a group of facts that are used to store information in cloud databases. The security policies offered must better reflect the idea of time-sensitive analysis. Depending on the time, the cloud data store can be used to access and store the encoded message or information. Although the CP-ABE method offers security, it has the drawback of keeping several copies of data side by side to make a single copy of the data, increasing the risk of data loss and increasing security risks. Numerous CP-ABE systems [2] have been put forth for use in a variety of contexts, including brief ciphertext and comprehensive security proofs. But none of the sophisticated access structures of the current CP-ABE techniques sufficiently handle the problem of decryption key size, which frequently turns into a disadvantage because of excessive resource usage. Scalability and efficiency issues arise in the majority of modern CP-ABE systems because the size of the decryption key increases as the number of characteristics increases. The forms of data protection technology that make up the CIA (Confidentiality, Integrity, and Availability) trinity are depicted in Fig.1.

- Confidentiality: It is the quality of something only being available to those who have been given permission. Unauthorized access to data is prevented by security measures.
- Integrity: Its quality is that it hasn't been changed by an unauthorized person. Integrity also includes maintaining the quality of data.
- Availability: It has the quality of being available and usable for a certain amount of time.

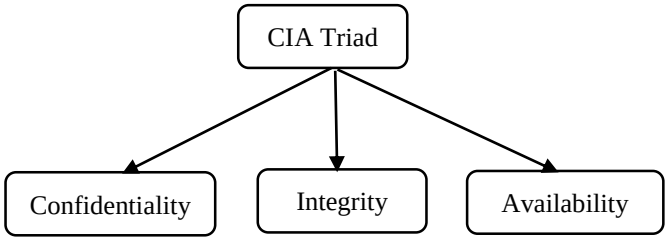

Fig.1. Data Protection methodologies

Nesrine et al. [3] proposed ID-based encryption, in which the content is first encoded and then kept on a cloud infrastructure server. Additionally, this idea provides access control, ensuring that the information may only be used by those who have authorization. With this strategy, unauthorized people aren't even able to view the data without the client's permission. Cloud security concerns are discussed by [4] before they offer a security architecture for the cloud that makes use of the Elliptical Curve Cryptography and Diffie Hellman Key Exchange algorithms. The whole model is broken down into four phases, the first of which is connection establishment, the second of which is account creation, the third of which is authentication, and the fourth of which is data sharing. For various objectives, further ABE systems are proposed. Chase [5] presented the initial multi-authority attribute-based encryption (ABE) architecture. Later, ABE techniques that supported partially disguised access structures were proposed by Nishide et al. [6], while Li and Chen [7] introduced an ABE method with quick decryption. A white-box traceable CP-ABE using monotonic access structures was created by Liu et al. [9], while Hinek et al. [8] addressed the important cloning problem in ABE.

Additionally, CP-ABE systems have been categorized according to their access architectures. Systems with AND gates and threshold gates for brief ciphertexts were first presented by Herranz et al. [10] and Li et al. [11], respectively. In addition, Li et al. [9] introduced the initial key-policy hierarchical-authority ABE using non-monotonic access structures (KP-HABE-NMaCS), which supports enormous universes, constant-size ciphertexts, and key delegation.

Moreover, they expanded their plan to allow for decryption to be outsourced and established a new security model that demonstrated selective security in the conventional approach. Despite these advancements, there are still a number of issues with traditional CP-ABE systems. Access control lists may

expose user identities since they are not explicitly encoded. Outsourcing Smart Health Records (SHRs) to the cloud also raises the possibility of manipulation. In order to solve this, Dutta et al. [12] proposed a policy-hiding CP-ABE (PHCA) method that includes cloud auditing and a constant decryption cost. At the same time, puncturable encryption (PE) was created so that recipients may update their decryption keys without getting in touch with the senders. On the basis of this concept, puncturable key-policy ABE (KP-PABE) and puncturable identity-based encryption (PIBE) were developed to provide fine-grained access revocation.

CP-ABE has been used by several researchers in IoT and healthcare settings. An RSA-enhanced encryption method for scalable patient record access control was presented by Sharma et al. [13]. A HAP-CP-ABE technique was created by Saravanan and Umamakeswari [14] for PHR systems that use HAP-based authentication. A reduced-size ciphertext multi-authority CP-ABE (RMA-CPABE) designed for fog-enabled IoT scenarios was presented by Chaudhary et al. [15]. ABE's suitability for cloud services was investigated by Kumar and Alphonse [16], who focused on privacy and fine-grained access control. According to Deshmukh et al. [18], the majority of healthcare systems use CP-ABE versions because they facilitate policy changing and concealment. Cheung and Newport [20] proposed dynamic policy updating to improve system security, whereas Catherine and Nargunam [19] proposed a structured ciphertext-policy approach for adaptable, traceable multi-user data sharing.

Despite offering robust fine-grained access control, CP-ABE still has some important drawbacks.

- The handling and revocation of attribute keys are complicated aspects of key management.
- Scalability: As the number of characteristics increases, so do the sizes of the key and ciphertext.
- Policy expressiveness: It is challenging to model intricate and layered access conditions.
- Privacy risks: sensitive user data may be made public by attribute authorities.
- Interoperability problems: Adoption in practical systems is hampered by a lack of standards.

In response, this study suggests a CP-ABE and RSA-based solution (CARSA) that offers fine-grained, privacy-preserving access control appropriate for cloud and healthcare applications, improves scalability, and lowers key management complexity. The necessity for fast data exchange, safe communication, and secrecy in delicate fields like healthcare and the Internet of Things is what drives the CARSA plan.

The paper's remaining sections are arranged as follows: We give pertinent background information on CP-ABE and its solutions in section 2. In section 3, basics of RSA algorithm and its applications are mentioned. Section 4outlines the methodology and approach used in developing the proposed CARSA solution for accurate access. Section 5 is presented with the results obtained from the experiments and analyzes the performance of the CARSA solution in making accurate access and providing explanations.

2. BACKGROUND AND PRELIMINARIES OF CP-ABE

The mathematical foundations and terminologies related to CP-ABE are covered in the next upcoming section.

2.1 ACCESS STRUCTURE AND ATTRIBUTES

According to [13], we specify the characteristic and the access policy. The collection of n attributes $A_1, A_2,\ldots A_n$ should be considered the attribute universe $U_A = \{A_1, A_2,\ldots A_n\}$. For ease of use, we represent it by subscript i the attribute or character set A_i

We refer to a user's attribute or character set as $A_s \subseteq U_A$, and the definition of the n-bit string $a_1, a_2,\ldots a_n$ connected to A_s is as follows:

$$a_i = \begin{cases} 1, & \text{if } A_i \in A_s \\ 0, & \text{if } A_i \notin A_s \end{cases}$$

For instance, the 4-bit string linked using A_s becomes 1101 if n=4 and $A_s=\{A_1,A_2,A_4\}$. In addition to the previous attribute set, we establish an access policy via P, with properties described in U_A. The definition of the n-bit string $b_1,b_2,\ldots b_n$ connected to P is as follows:

$$b_i = \begin{cases} 1, & \text{if } A_i \in P \\ 0, & \text{if } A_i \notin P \end{cases}$$

For instance, assume that the 4-bit string linked with P becomes 1010 if n=4 and $P=\{A_1,A_3\}$.

Definition 1: The access structure $P= \{b_1,b_2,\ldots b_n\}$ is satisfied by the attribute set $A_s=\{a_1,a_2,\ldots a_n\}$if we have $a_i \geq b_i$ for all i=1 *to* n. For the abbreviation of A_s fulfilling P, we consider $P \subseteq A_s$.

2.2 DEFINITION OF CP-ABE

A CP-ABE technique is composed of the following four algorithms: Setup, Encrypt, KeyGen, and Decrypt. The following defines these algorithms [14]:

- ***Setup*:** This method uses the collection of characteristics $U_A= \{A_1,A_2,\ldots A_n\}$ and the security parameter Λ as inputs to generate a master public key M_{PK} and a master secret key M_{SK}.
- ***Encryption:*** It requires three inputs: plaintext M_p, the M_{PK}, and an access policy P. A ciphertext CT is produced using the encryption method $E[P, M_p]$.
- ***KeyGen:*** The inputs of the algorithm are an attribute set A_s, the M_{PK}, and the M_{SK}. The user secret key (decryption key) S_k corresponding to A_s is then produced using the key generation procedure.
- ***Decryption:*** It accepts as inputs a ciphertext CT produced by an access policy P, the secret key S_k belonging to the attribute set A_s, and outputs the M_P.

A CP-ABE scheme must satisfy the following property: if the attribute set A_s associated with a secret key S_k satisfies the access policy P (i.e., $P \subseteq A_s$), then the decryption algorithm always recovers the original plaintext M_p from a ciphertext $E[P, M_p]$, using ($M_{PK,}$ M_{SK}). Otherwise, when $P \nsubseteq A_s$, the secret key S_k cannot decrypt the ciphertext, and the plaintext remains inaccessible.

3. BASICS OF RSA ALGORITHM

A popular public-key cryptographic technique for digital signatures and secure communication is called RSA (Rivest–Shamir–Adleman). The computational complexity of factoring huge composite numbers to their prime elements is the foundation of their security. Such factorization is practically impractical, which guarantees the security of encrypted messages. A crucial component of public-key cryptography, RSA consists of three primary steps:

- Making a pair of public and private keys for both encryption and decryption is known as key generation.
- Encryption: Using the recipient's public key, data is transformed from plaintext to ciphertext.
- Decryption: Using the private key to get the original text from the ciphertext.

3.1 KEY GENERATION

The five integers *P*, *Q*, *N*, *E*, and *D* must be calculated in order for the RSA public key algorithm to function properly. Algorithm 1 shows that the prime values *P* and *Q* are both enormous. The N is created by multiplying *P* and *Q*. Decide on an optimistic integer *E* such that gcd(*E*, Φ(*N*)) = 1 and then calculate $1 < E < \Phi(N)$.The user chooses the encryption key represented by the *E*. The key for decryption is the *D*, which is computed by utilizing the numbers. The private key or decryption key is then calculated as follows: $1<D<\Phi(N)$, so that $E.D \ mod \Phi(N)=1$. By calculating the multiplicative inverse of the encryption key *E*, one can derive the decryption key *D*.

Algorithm 1: Key Generation Algorithm

Input	*P and Q – two large prime numbers*
Output	K_{pub} *– public key and* K_{pri} *– private key*
1. Compute the modulus $N = P*Q$; *2.* Calculate the totient $\Phi(N)=(P-1)*(Q-1)$; *3.* Select an integer E such that $1 < E < \Phi(N)$, $gcd(E, \Phi(N))=1$ *4.* Compute the private exponent *D* as the modular inverse of *E* modulo $\Phi(N)$ i.e., $D.E \equiv \Phi(N)$	

3.2 ENCRYPTION PROCESS

Through Eq.(1), which requires the sender to conduct modular exponentiation, the Encryption algorithm process can be calculated. Get the heir's public key (*E, N*) first, as well as the plaintext that has to be delivered, which is denoted by *M*. After receiving *E*, *N*, and *M*, convert the actual text into cipher- or coded text. You then transmit the receiver the encrypted text across the channel.

$$C= M^E \ mod \ N \tag{1}$$

3.3 DECRYPTION PROCESS

Eq.(2) denotes the decryption process which must carry out the opposite action to that of the encryption process. The recipient must use his private key (*D,N*) to decode the data in order to retrieve the original plaintext. To achieve this, compute the modular exponential function of *c* with regard to the modulus N in order to decrypt the cipher text that was sent by the sender.

$$M= C^D \ mod \ N \tag{2}$$

4. ALGORITHM FOR THE PROPOSED SYSTEM

The four operational phases of the proposed CARSA approach are data encryption, setup, data decryption, and access control policy enforcement. An outline of the cloud-based data that is outsourced and the security of the upgraded CP-ABE scheme that incorporates RSA are given in this part. The four primary components of the CARSA architecture are the data owner, data consumers, cloud server, & the data itself, as shown in Fig.2. The creation, distribution, and revocation of cryptographic keys for other system entities are under the purview of the data owner. By assigning attribute sets to data items according to their attributes, the data owner also establishes access controls for certain data depending on user roles. Requests for access to cloud-stored data can be made by data users, who can then decode the ciphertext and obtain the desired data in accordance with the established access regulations using the keys provided by the data owner. The cloud server helps maintain updated ciphertexts & keys as needed and is in charge of safely storing the encrypted data.

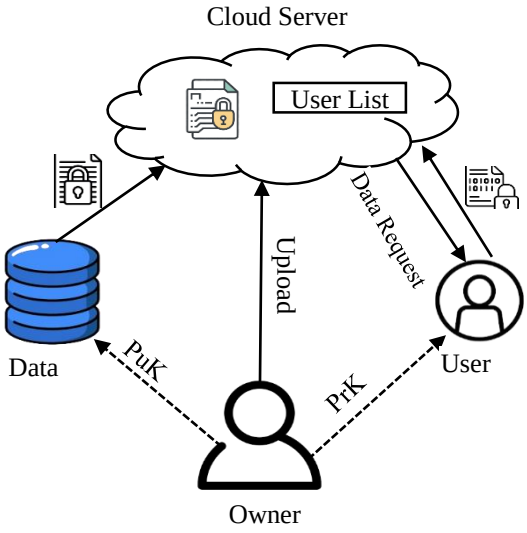

Fig.2. System model for CARSA method

4.1 SETUP PHASE

In CARSA algorithm, there is a setup phase involved to establish the necessary cryptographic parameters and generate the required keys. In Algorithm 2, master public key and master secret key are generated. Here's an algorithm of the setup phase in CARSA:

Algorithm 2: Setup algorithm
Input: Security parameter Λ, attribute universe $UA=\{A_1,A_2,\ldots,A_n\}$
Output: Master public key *MPK*, master secret key *MSK*, RSA key pair (*PK,SK*)
Use Algorithm 1 to generate the RSA key pair (*PK,SK*)
CP-ABE Master Key Generation
Choose a cyclic group *G* of prime order *p* with generator *g*
Select a random master secret $\alpha \in Z_p$
Compute g^{α} for inclusion in public parameters
For each attribute $A_i \in UA$:

Select a random secret $k_i \in Zp$
Compute the public component $T_i=g^{k}_{i}$
Include k_i in MSK
Hash Function Selection for Policy Binding
Select the following three pre-image – resistance hashing operations H_1, H_2, H_3.

$$H_1 : \alpha^* \rightarrow \alpha^{\rho}$$

$$H_2 : \alpha^* \rightarrow \alpha^{l_\sigma}$$

$$H_3 : \alpha^* \rightarrow \alpha^{l_m}$$

Compute the public parameters for CARSA as follows*:*
Select random $g_r \in G$ such that $2<g_r<N-1$ and $gcd(g_r,N)=1$
Compute $V_u = g_r^{v_u}$, $W = g_r^{x}$, $X = g_r^{k}$, *where* $v_u = \prod_{A_i \in U_A} q_i$
Return Master public key MPK, master secret key MSK, RSA key pair (*PK*,*SK*)
end

4.2 ENCRYPTION PHASE

An access policy P, the master public key M_{PK}, and the original message M_p are the inputs for this algorithm. The result of the encryption algorithm is ciphertext CT.

$$CT = E(\sigma_m, H_1(P, M_p, \sigma_m)),$$

$$H_3(\sigma_m) \oplus M_p, S_m = H_1(\sigma_m, M_p))$$

Using the hash output, let σ_m represent a secret of random value. In order to confirm that the derived plaintext M_p is legitimate, we additionally calculate the signature $S_m = H_1(\sigma_m, M_p)$ using the random secret σ_m. The proposed system encryption algorithm is depicted in Algorithm 3 as follows:

Algorithm 3: Encryption algorithm

Input: Access policy *P*, master public key *MPK*, plaintext *Mp*
Output: Ciphertext *CT*
Pick a random number $\sigma_m \in \boldsymbol{\alpha^{l_\sigma}}$
Compute the random value r^m
Compute the session key $K_m = V_u^{\frac{r_m e_U}{e_P}} = g^{r_m d_p}$
Compute the ciphertext *CT* with the help of $X_m, R_m, C\sigma_m, C_m, S_m$
Return *CT*

4.3 ACCESS CONTROL PHASE

To demonstrate our concept, we initiate by presenting a straightforward instance of a monotone policy. A monotone access policy, when represented in its Disjunctive Normal Form (DNF), inherently comprises individual access structures based on logical AND (∧) conditions. Consequently, the amalgamation of these AND (∧) based access policies through the inclusion of OR operators between them inherently results in a potent and expressive monotone access policy [17]. To enhance access control granularity, we introduce a distinct attribute called "privilege" as an extended leaf (EL) node within the Access Control Policy Tree (ACPT). This attribute serves the purpose of identifying whether a role possesses read or write privileges. Consider the below policy which governs access to patient data, ensuring that only authorized hospital staff and patients can retrieve and interact with the data. Take into account a policy centered around patients, denoted as P = (Hspt-1 ∧ Doctor) OR (Institution-1 ∧ Professor) OR (Institution -1 ∧ Student) OR (Insurance Organization-1 ∧ Insurance-agent).

The policy is overseen by the hosting hospital and possesses the capability to undergo modifications by duly authorized administrators. In practicality, this policy holds the potential for alteration at any given time. For instance, a scenario might arise where the senior nurse gains authorization to access the diagnosis file in order to compile a summarized report. In such an instance, the data owner is required to augment the aforementioned policy structure by incorporating the "nurse" role along with its corresponding attributes, guided by logical rules that delineate the sanctioned access to the diagnosis file.

Apart from the policy update, the file that was previously encrypted based on the earlier policy must be retrieved from the cloud. Subsequently, it will undergo a process of decryption and subsequent re-encryption in alignment with the new policy. Following this, the re-encrypted file will be uploaded back to the cloud. This task can be quite laborious, particularly in cases involving substantial volumes of data, and is further compounded by the heightened likelihood of frequent policy modifications.

4.3.1 Access Control Policy:

Here's a high-level algorithm for enforcing an access control policy using CARSA methodology:

Inputs:

User attributes U_A: List of attributes possessed by the user $U_A = \{A_1, A_2, \ldots A_n\}$.

Access Policy P: Boolean expression representing the required attributes and their values for access.

Encrypted Document CT: Document encrypted using CP-ABE.

Output: *Access Granted P'*: Boolean value indicating whether access is granted or denied.

Procedure:

i) Extract attributes from the user's attribute-based key or identity.
ii) Utilize the user's characteristics to assess the access policy expression:
   a. Traverse the access policy expression.
   b. For each attribute condition:
iii) Check if the user possesses the required attribute.
iv) Return Access Denied if the user lacks the characteristic.
v) Go to the next step if all attribute requirements are met by the access policy.
vi) Attempt to decrypt the encrypted document using the user's attribute-based key or identity:
   a. If decryption is successful, return Access Granted.
   b. If decryption fails, return Access Denied.

The above algorithm provides an outline for enforcing an access control policy using CARSA. It assumes that the necessary keys, attributes, and the access policy expression are available for evaluation. The actual implementation details may vary depending on the specific scheme being used and the programming language or cryptographic library employed. The algorithm above focuses on the access control policy evaluation and document decryption part within the CARSA framework.

4.3.2 Access Control Policy Update:

To finalize the process of policy updating, two distinct tasks need to be executed: policy updating itself and the subsequent re-encryption of files. With this objective in mind, we introduce a policy updating algorithm. An adaptable and secure approach to policy updating must be established, enabling data owners or administrators to effectively manage attributes (such as addition, modification, or removal) within the policies stored on a cloud server. We have devised a policy updating algorithm designed to facilitate the modification of access policies within a cloud environment. This algorithm is geared towards minimizing computation and communication expenses while affording data owners the flexibility to update policies at their convenience, regardless of their location. Algorithm 4 represents the policy updating algorithm with inputs as non-leaf nodes (NLN') which represents user role node and threshold gate node, current leaf node (CLN), typeOfUpdate and policyID. Output is the new access policy (NP').

Algorithm 4: Policy Updation algorithm

Inputs: NLN', CLN, typeOfUpdate, PolicyID
Output: NP'
Procedure:

Assign *ACP* where pid=policyID
if (typeOfUpdate == delete)
Assign NP' = ACP.replace((CLN),"")
else if(typeOfUpdate == edit)
Assign NP' = ACP.replace((CLN),"NLN'")
else if (typeOfUpdate == add)
Assign NP' = ACP.insert(NLN')
policy=NP'
Assign and get proper fileID
foreach (fileID as id)
id.reencrypt(NP')
end foreach
if (policy_is_verified(NP'))
return TRUE
else
return FALSE

4.4 KEY – GENERATION PHASE

The user secret key is generated using the following steps:

Inputs: *MPK* and *MSK*

Output: The user secret key $S_k = \{S_1, S_2\}$ that corresponds to the user characteristics A_s

1. Compute $t_A = \prod_{j=1}^{n+1} q_j^{a_j}$, so that $a_j = 1$ *if* $A_j \in A_s$
2. Randomly pick two random numbers c_u and d_u and then find e_u.
3. Finally find the key values $S_1 = e_u + k_2 d_u$ and $S_2 = c_u - k_1 d_u (mod\ \Phi(N))$.

4.5 DECRYPTION PHASE

In this stage, we go through the procedures used for decryption.

Inputs: Ciphertext CT matching a certain access policy P and the user's secret key S_k matching the user characteristic A_s.

Output: The original message M_p

1. Calculate $K_m = V_u^{\frac{r_m e_U}{e_P}} = g^{r_m d_p}$
2. Calculate $\sigma_m^{'} = H_2 \oplus (C \cdot \sigma_m)$ and $M_p^{'} = C_m \oplus H_3(\sigma_m^{'})$
3. Finally, determine if the $S_m = H_1\left(\sigma_m^{'}, M_p^{'}\right)$ condition is true or false to determine if the signatures match. If it is true, output $M_p^{'}$; else, output NULL.

5. SECURITY ANALYSIS

The major data owner may authorize the granting of access to additional subordinate users following the first upload of data to the cloud. In order to access and, if permitted, alter the data, the cloud server provides the public key & secret key to both the primary owner and subordinate users. By submitting their secret key, which combines their identity & ownership information, a subordinate user has access to encrypted material that the primary owner uploads to the cloud. For the user & subordinate user to have access, the primary owner has to supply both the access key and the RSA key. The same is true when someone requests data; access is only allowed if the policy's requirements are met after the policy has been checked. The user can't get the data back from the cloud if the owner of the data revokes access.

Attackers usually try to intercept interactions between a user and the data owner in man-in-the-middle (MITM) attacks. In this situation, the two parties are unaware that messages are being transmitted between them. Critical data, such as the answer to randomly generated security challenges plus the secret key needed for decryption, would be necessary for the attacker to get access to cloud-stored data. Data is therefore shielded from MITM attacks as the attacker cannot get beyond the security architecture.

Additionally, the system's security keys are extremely difficult to hack. The RSA algorithm, an asymmetric cryptography method, is the mainstay of the proposed structure. Consequently, in order to access the data, an attacker would require both of the decryption keys. The access key, which is set up in accordance with the access policy and contains the user or owner attributes together with a randomized identity, would still be necessary for the attacker even if the RSA key was compromised. Consequently, the proposed paradigm offers more security than symmetric key encryption schemes like EECDH. The proposed scheme's complex security key structure guarantees that data saved in the cloud is successfully shielded from a variety of threats.

6. EXPERIMENTAL RESULTS AND COMMENTS

This section includes simulation experiments to assess the proposed CARSA scheme's performance. The Pairing-Based Cryptography (PBC) library and the Open Secure Socket Layer (OpenSSL) package are used to implement all methods on a Linux

computer with an Intel(R) Core(TM) i5-6200U CPU operating at 2.4 GHz and 8 GB of RAM.

An evaluation and comparison of the proposed improved CARSA scheme with current methods for adaptable and safe public auditing of cloud user data are conducted. The following are the assessment metrics: encryption time, access time, decryption time, and overall execution time. The time (in milliseconds) needed for each operation across various key lengths is shown in Table.1. Ten trials with key sizes of 8, 16, 32, 64, 128 and 256 bits provide the results. The proposed CARSA technique needs 1610 ms for encryption and 1098 ms for decryption for a 256-bit key length. Processing a 256-bit key length takes 3254 milliseconds in total. The Fig.3 provides a graphic representation of the proposed CARSA scheme's encryption, decryption, and execution time performance, highlighting its effectiveness and appropriateness for safe cloud data management.

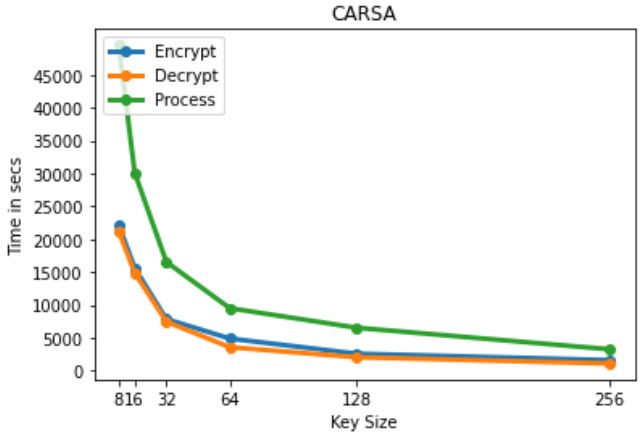

Fig.3. Encryption, Decryption and Execution time for proposed CARSA method

To measure the execution time, we take into account the CARSA processes associated with personal and shared outsourced files. Our data set includes both text and media files, varying in size from 1 KB to 15 KB. For each experiment, we conduct few runs to obtain reliable results. To ensure accuracy, we flush the cache between each test. The reported results are the average of these runs, providing a representative measure of the response time for the given operations in the Table.1.

The performance evaluation of the proposed CARSA encryption method is shown in the Table.1 for a range of file sizes, from tiny (1 KB) to big (50 MB), and for several file kinds, including text, PDF, and image. Access time, encryption time, and decryption time are the important metrics. As anticipated, the processing burden associated with greater data is reflected in all three measures, which rise with file size. Because of their simplicity, text files have the quickest encryption and decryption times, followed by PDFs and photos, which take somewhat long, and larger text files, which take the longest because of their size and complexity. For smaller files, decryption durations are often shorter than encryption times; but, for bigger files, they approach comparable values. This is in line with the CARSA scheme's extra processing for randomization during encryption and access policy enforcement. Access time steadily rises with file size but is comparatively low when compared to encryption and decryption, suggesting lightweight policy verification. Overall, the findings show that CARSA is appropriate for cloud-based systems with heterogeneous data as it effectively enables safe encryption, decryption, and fine-grained access control across various file types and sizes. The performance comparison between the proposed CARSA scheme and the current ABE approaches is shown in Table.2. For different key sizes, metrics include throughput, ciphertext size, execution time, encryption time, and decryption time. The outcomes show how effective and scalable CARSA is at protecting data stored in the cloud.

Decoding the ciphertext and accessing its contents will only be possible if the user's attribute set along with external variables meet the requirements. We may thus draw the conclusion that the method can provide versatile and fine-grained access management for urgent data on public clouds based on the performance study of CP-ABE and the RSA method.

Table.1. Response time (seconds)

File Type	File Size	Encryption Time (ms)	Decryption Time (ms)	Access Time (ms)
Text	1 KB	0.13	0.02	0.11
Text	2 KB	0.15	0.04	0.15
Text	4 KB	0.21	0.08	0.16
Text	8 KB	0.26	0.15	0.19
Text	15 KB	0.30	0.17	0.20
PDF	50 KB	0.55	0.25	0.30
PDF	100 KB	0.95	0.48	0.42
PDF	500 KB	3.20	1.85	1.10
Image	1 MB	6.50	3.40	1.90
Image	5 MB	28.00	14.50	8.20
Text	10 MB	56.00	29.00	15.00
Text	50 MB	280.00	145.00	75.00

Table.2. Analysis of the upgraded CARSA method's performance quantitatively in terms of encryption time, decryption time, and processing time against existing CP-ABE.

Key Size (bits)	Encryption Time (ms)		Decryption Time (ms)		Execution Time (ms)		Cipher Text Size (KB)	Throughput (KB/ms)
	CARSA	Existing ABE [9]	CARSA	Existing ABE [9]	CARSA	Existing ABE [9]	CARSA	CARSA
8	52	60	38	45	90	110	1.2	0.95
16	105	120	77	90	185	210	2.4	1.02
32	210	240	155	180	365	420	4.8	1.05
64	425	480	310	360	780	840	9.6	1.10
128	850	960	620	720	1470	1680	19.2	1.12
256	1610	1691	1098	1134	3254	3380	38.4	1.18
512	3250	3400	2200	2260	6000	6400	76.8	1.20
1024	6500	6800	4400	4520	12000	12500	153.6	1.22

7. CONCLUSION AND FUTURE WORK

According to this study, the CARSA technique enables fine-grained access control, in which a user's ability to access encrypted data is based on particular characteristics they possess. This attribute-based access control allows flexible and adjustable

data sharing procedures. Additionally, it offers a safe way to send and receive encrypted communications across unprotected channels, guaranteeing the communication's integrity and secrecy. This article primarily focuses on safe data storage in public clouds. It offers the ability for time-sensitive data access control at a finer level. By combining the CP-ABE and RSA algorithms as a proposed methodology named CARSA, this may be accomplished. In accordance with the clearly specified access policy regarding attribute and releasing time, the data owners can use this method to determine which users are permitted to access data and offer necessary access rights releasing time points. The data holder encodes the data using the CP-ABE approach and transmits the encrypted message to the cloud system. The user's important encoded data is decoded using the RSA method. Total processing time for encryption and decryption was 3254 milliseconds. Simulation is used to evaluate the effectiveness of the proposed and current approaches in the cloud. This method is ideal for large-scale cloud storage access control systems. The RSA algorithm can be enhanced in the future to boost performance by increasing the level of security for data that is outsourced. CARSA is efficient for encrypting and decrypting relatively different types of data. It is commonly used for encrypting symmetric keys that are then used for encrypting larger amounts of data using symmetric encryption algorithms. Additionally, it ensures the security and integrity of the connection by offering a safe way to exchange encrypted communications via unsecure channels. In the future, we will focus on minimizing the number of decryption tests needed for approved receivers with extra redundant properties not covered by the ciphertext access policy.

REFERENCES

[1] S. Xue and C. Ren, "Security Protection of System Sharing Data with Improved CP-ABE Encryption Algorithm Under Cloud Computing Environment", *Automatic Control and Computer Sciences*, Vol. 53, No. 4, pp. 342-350, 2019.

[2] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based Encryption for Fine-Grained Access Control of Encrypted Data", *Proceedings of International Conference on Computer and Communications Security*, pp. 89-98, 2006.

[3] N. Kaaniche, A. Boudguiga and M. Laurent, "ID based Cryptography for Secure Cloud Data Storage", *Proceedings of International Conference on Cloud Computing*, pp. 1-6, 2013.

[4] R. Ganesan, "Data Security in Cloud Architecture based on Diffie Hellman and Elliptical Curve Cryptography", *Cryptology ePrint Archive*, pp. 1-5, 2014.

[5] M. Chase, "Multi-Authority Attribute based Encryption", *Theory of Cryptography Conference*, pp. 515-534, 2007.

[6] T. Nishide, K.Yoneyama and K. Ohta, "Attribute-based Encryption with Partially Hidden Encryptor-Specified Access Structures", *Applied Cryptography and Network Security*, pp. 111-129, 2008.

[7] Z. Li and X. Chen, "Attribute-based Encryption with Fast Decryption on Prime Order Groups", *Journal of Computer Applications*, Vol. 36, No. 3, pp. 1-16, 2016.

[8] M.J. Hinek, S. Jiang, R. Safavi-Naini and S.F. Shahandashti, "Attribute-based Encryption without Key Cloning", *International Journal of Applied Cryptography*, Vol. 2, No. 3, pp. 250-270, 2012.

[9] Z. Liu, Z. Cao and D.S. Wong, "White-Box Traceable Ciphertext-Policy Attribute-based Encryption Supporting any Monotone Access Structures", *IEEE Transactions on Information Forensics and Security*, Vol. 8, No. 1, pp. 76-88, 2012.

[10] J. Herranz, F. Laguillaumie and C. Ràfols, "Constant Size Ciphertexts in Threshold Attribute-based Encryption", *Public Key Cryptography*, pp. 19-34, 2010.

[11] C. Li, Q. Shen, Z. Xie, J. Dong, X. Feng, Y. Fang and Z. Wu, "Hierarchical and Non-Monotonic Key-Policy Attribute-based Encryption and its Application", *Information Sciences*, Vol. 611, pp. 591-627, 2022.

[12] P. Dutta, W. Susilo, D.H. Duong and P.S. Roy, "Puncturable Identity-based and Attribute-based Encryption from lattices", *Theoretical Computer Science*, Vol. 929, pp. 18-38, 2022.

[13] K. Sharma, A. Agrawal, D. Pandey, R.A. Khan and S.K. Dinkar, "RSA based Encryption Approach for Preserving Confidentiality of Big Data", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 5, pp. 2088-2097, 2022.

[14] N. Saravanan and A. Umamakeswari, "HAP-CP-ABE based Encryption Technique with Hashed Access Policy based Authentication Scheme for Privacy Preserving of PHR", *Microprocessors and Microsystems*, Vol. 80, pp. 10-32, 2021.

[15] C.K. Chaudhary, R. Sarma and F.A. Barbhuiya, "RMA-CPABE: A Multi-Authority CPABE Scheme with Reduced Ciphertext Size for IoT Devices", *Future Generation Computer Systems*, Vol. 138, pp. 226-242, 2023.

[16] P. Kumar and P.J.A. Alphonse, "Attribute based Encryption in Cloud Computing: A Survey, Gap Analysis, and Future Directions", *Journal of Network and Computer Applications*, Vol. 108, pp. 37-52, 2018.

[17] R. Imam, K. Kumar, S.M. Raza, R. Sadaf, F. Anwer, N. Fatima, M. Nadeem, M. Abbas and O. Rahman, "A Systematic Literature Review of Attribute based Encryption in Health Services", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 9, pp. 6743-6774, 2022.

[18] J.Y. Deshmukh, S.K. Yadav and G.M. Bhandari, "Attribute-based Encryption Mechanism with Privacy-Preserving Approach in Cloud Computing", *Materials Today: Proceedings*, Vol. 80, pp. 1786-1791, 2023.

[19] V.R. Catherine and A.S. Nargunam, "Multi Authority Ciphertext-Policy Attribute-based Encryption for Security Enhancement in Cloud Storage Unit", *Sustainable Energy Technologies and Assessments*, Vol. 53, pp. 10-25, 2022.

[20] L. Cheung and C. Newport, "Provably Secure Ciphertext Policy ABE", *Proceedings of International Conference on Computer and Communications Security*, pp. 456-465, 2007.