

**DESIGN AND DEVELOPMENT OF AN ATTACK-RESISTANT
IOT CLOUD CONVERGENCE ALGORITHM USING
ATTRIBUTE-BASED ENCRYPTION**

**Deepak D. Sapkal, Ritesh.V. Patil, Parikshit.N. Mahalle,
Surendra A. Mahajan, Lalit V. Patil**

Research Scholar, Smt. Kashibai Navale College Engineering (SKNCOE),
Pune, Maharashtra, India.

PVG'S College Of Engineering, Technology and Management, Pune,
Maharashtra, India.

Pune District Education Association's College of Engineering, Pune,
Maharashtra, India.

Vishwakarma Institute of Technology, Pune, Maharashtra, India.

PVG'S College Of Engineering, Technology and Management, Pune,
Maharashtra, India.

Smt. Kashibai Navale College Engineering (SKNCOE), Pune, Maharashtra,
India.

ddsapkal@gmail.com, rvpatil3475@yahoo.com, aalborg.pnm@gmail.com,
sa_mahajan@yahoo.com, lvpatil@sinhgad.edu

Abstract

As the number of Internet of Things (IoT) devices grows quickly, it becomes harder to keep data handling and communication safe in cloud settings. This paper describes how an attack-proof IoT cloud convergence method was designed and built. It uses Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to provide strong security and fine-grained, attribute-based access control. The suggested framework uses CP-ABE based on the Bethencourt-Sahai-Waters (BSW) method to make key generation, sharing, encryption, and decoding safe and user-attribute-aligned. This makes sure that only authorised users can access critical data. To make sure the encryption is strong and works well, the method uses the JPBC library to perform pairing-based cryptography processes. Using attribute-based registration and login, the user module makes sure that users are who they say they are and that they have the right permissions. The system handles files by encrypting them with lightweight AES for privacy, hashing them with SHA-256 for integrity checks, and letting you securely share and download files with reporting support. The Third Party Auditor (TPA) tool also makes sure that files are real and manages user termination to stop people from getting in without permission after a key has

been compromised or a characteristic has been changed. A performance study shows that the suggested method works by comparing how much time and memory are used for decrypting both insourced and outsourced files of different sizes. The results show that the system strikes a good balance between security and speed. This means that it can be used in real-life IoT-cloud situations where resources are limited and changing access control is important. The comparison study shows that attribute-based encryption is better at protecting against different types of threats while still having a reasonable amount of extra work to do.

Keywords: IoT Security, Attribute-Based Encryption, Ciphertext-Policy ABE, Cloud Data Protection, Attack-Resistant Algorithm

I. Introduction

The fast growth of Internet of Things (IoT) devices has changed how data is gathered, handled, and used in many areas, including smart cities, healthcare, home automation, and industrial automation. IoT devices create huge amounts of private data that needs to be stored and managed safely. Cloud computing offers storage and computing power that can be scaled up or down to support these large-scale IoT operations. But when IoT and cloud computing come together, they create big security and privacy problems. This is mostly because IoT devices are spread out, users are different, and access needs to change all the time. Making sure that safe and fine-grained data access control can work in these kinds of settings is still a very important study problem. Most traditional encryption methods use identity- or role-based access controls, which aren't flexible enough to work with the complicated and changing attribute-based rules that are needed in IoT-cloud ecosystems [1]. Because the cloud has multiple tenants and can be attacked from both inside and outside, it is very important to create security systems that can handle a wide range of attack methods. Current methods often have trouble finding the right mix between security, processing speed, and scaling, which is especially hard for IoT devices that don't have a lot of resources. The study's goal is to solve these problems by creating an IoT cloud convergence method based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE) that can't be attacked [2]. CP-ABE lets you control who can see protected data by linking it to access rules that are based on user traits instead of set names.

The data can only be decrypted by users whose characteristics match the encryption policy. This makes the system safer and more flexible. The solution uses the CP-ABE method by Bethencourt, Sahai, and Waters (BSW), which uses pairing-based cryptography to give strong security guarantees. The suggested framework includes several important parts, such as attribute-based user registration and login, key creation that is matched with user characteristics, lightweight AES encryption for effective file protection, and SHA-256 hashing for verifying file integrity [3]. A special section called Third Party Auditor (TPA) checks the integrity of the data and lets users be removed, which stops people from getting in without permission after changing attributes or having their keys stolen. This complete design not only keeps data safe and secure, but it also lets cloud-connected IoT systems control access in a way

that is both scalable and dynamic. Cyberattacks on IoT and cloud systems are getting smarter, which is why an attack-resistant design is so important. Some of these are insider threats, unauthorised data access, and conspiracy attacks, all of which can put private information at risk and stop services from working [4]. By combining CP-ABE with efficient cryptographic operations and system-level tracking, the suggested solution offers strong defences while keeping the computing load doable, making it ideal for IoT devices that don't have a lot of resources. To test how well the method works, we look at how much time and memory it uses for different file sizes in both insource and external decoding situations [5].

II. Related Work

There has been a lot of research done on the security of connecting IoT devices to the cloud because it is so important to keep private data safe. A lot of people have used traditional encryption methods to keep their data and communication routes safe, like symmetric key cryptography and public key infrastructure (PKI). But these methods often can't provide the fine-grained access control and scaling that are needed for IoT settings with a lot of different types of devices. Identity-Based Encryption (IBE) was created to help with some of these problems by connecting encryption keys to users' names. However, it is still not flexible enough to support access rules that change based on changeable attributes [6]. Attribute-Based Encryption (ABE), and especially Ciphertext-Policy Attribute-Based Encryption (CP-ABE), has become a potential way to use cryptography to make cloud computing and Internet of Things systems more secure and flexible in how users can access them. The CP-ABE method was first suggested by Bethencourt et al. [7]. It lets data owners set access rules that are contained in ciphertexts, making sure that only users with the right characteristics can decrypt the data. Several papers have improved CP-ABE for use in real-world IoT situations by making key generation and encryption more efficient to work with the limited resources that IoT devices usually have [8,9]. Additionally, experts have looked into the problem of removal of users and changing trait changes in ABE-based systems. For example, access control methods that use proxy re-encryption and sharing systems make it easier to change rules, but they often come with extra processing costs [10]. To fix this, lightweight encryption methods like AES and ABE have been mixed in hybrid encryption systems to make them more efficient without lowering security [11]. To improve faith in cloud settings, Third Party Auditors (TPAs) have also been looked at as a way to outsource data verification and security checks. TPAs can check the accuracy of data without seeing the code underneath, which protects privacy [12]. Table 1 summarizes methodologies, features, limitations, and IoT-cloud contributions. Also, pairing-based encryption tools like JPBC have made it easier to build complicated ABE methods on limited devices, which makes them useful for use in IoT-cloud systems [13].

Table 1: Summary of Related Work

Methodology	Key Features	Limitations	Contribution to IoT-Cloud Security
Ciphertext-Policy Attribute-Based Encryption (CP-ABE)	Fine-grained access control using attribute policies	High computational cost for large policies	Foundational CP-ABE scheme enabling attribute-based access control
Attribute-Based Encryption (ABE) [14]	Key generation based on user attributes	Limited policy expressiveness	Introduced ABE concept for flexible encryption in distributed systems
CP-ABE with improved efficiency	Reduced key size and encryption overhead	Complex key management	Enhanced CP-ABE scalability for practical use in constrained devices
Lightweight ABE for IoT [15]	Optimized encryption for resource-limited IoT devices	Limited support for dynamic attribute updates	Adapted ABE for IoT environments with efficiency improvements
User revocation in CP-ABE [16]	Proxy re-encryption and key delegation	Increased computational overhead	Addressed dynamic user revocation in ABE systems
Hybrid encryption combining CP-ABE & AES	Efficient file encryption with attribute control	Additional complexity in key synchronization	Improved encryption efficiency by integrating symmetric encryption
Third Party Auditor (TPA) for data integrity [17]	Outsourced verification with privacy preservation	Trust issues with third parties	Enabled secure data auditing without revealing content
JPBC library-based implementation [18]	Efficient pairing-based cryptography	Limited hardware acceleration support	Practical implementation of pairing-based ABE schemes on constrained devices
Attack-resistant ABE system [19]	Resistance to collusion and attribute forgery	Increased key generation time	Enhanced security against advanced attack vectors in cloud-IoT systems

Dynamic attribute update and revocation	Real-time policy updates and revocation	High communication overhead	Improved flexibility for dynamic user attribute management
IoT-cloud convergence security framework	Integration of ABE with cloud data management	Scalability issues with massive user base	Provided a framework for secure IoT-cloud data operations
Lightweight encryption & integrity verification	Combined AES encryption with SHA-256 hashing	Limited analysis of computational overhead	Balanced security and performance in IoT-cloud file operations

III. Methodology

A. Key Generation Module: Attribute Based Key Generation

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme, specifically based on the Bethencourt-Sahai-Waters (BSW) ABE scheme. This scheme allows for fine-grained access control over encrypted data, where access policies are defined over attributes, and only users with attributes satisfying the policy can decrypt the data. The CP-ABE is organized into several key components:

- Setup: Initializes the public parameters and master secret key.
- Key Generation: Generates private keys for users based on their attributes.
- Delegation: Allows for the delegation of keys to users with a subset of attributes.
- Encryption: Encrypts data under a specific access policy.
- Decryption: Decrypts data if the user's attributes satisfy the access policy.

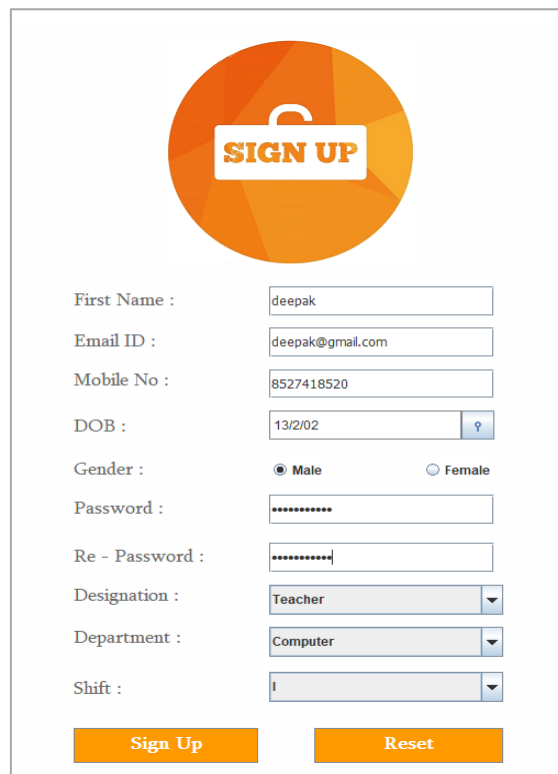
The implementation relies on Pairing-Based Cryptography (using the JPBC library) to achieve the cryptographic operations required for CP-ABE.

B. User Module

1. Registration (Attribute based registration)

The suggested IoT cloud convergence method uses attribute-based registration to make the user registration process more secure and easier to control who can see what. Instead of using standard identity-based registration alone, this method links user attributes—like jobs, rights, or organisational memberships—with the registration details. When users register, they give the system information about themselves, which is then checked and kept safely. These characteristics are used to make secure keys that are specific to each user's permissions. This fine-grained attribute binding makes sure that encrypted data can only be accessed by authorised people whose characteristics match. Attribute-based registration also makes it easier to handle users in IoT-cloud settings in a way that is dynamic and scalable. Figure 1

shows secure attribute-based user registration process flow. It does this by allowing flexible access rules that can change as user jobs and powers do without compromising security.

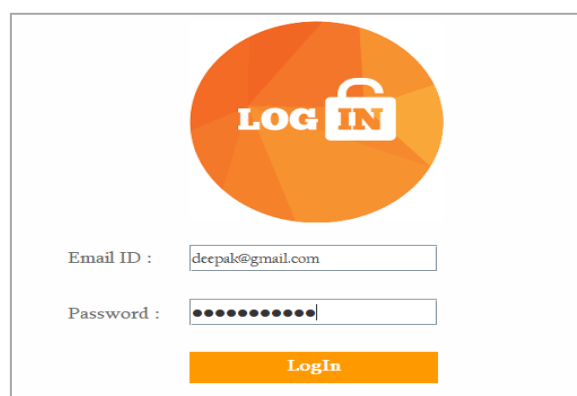


The registration form features a large orange circular icon with a white padlock and the text "SIGN UP" at the top. Below this, the form contains several input fields: "First Name" (filled with "deepak"), "Email ID" (filled with "deepak@gmail.com"), "Mobile No" (filled with "8527418520"), "DOB" (filled with "13/2/02" and a calendar icon), "Gender" (radio buttons for "Male" and "Female", with "Male" selected), "Password" (masked with dots), "Re - Password" (masked with dots), "Designation" (dropdown menu with "Teacher" selected), "Department" (dropdown menu with "Computer" selected), and "Shift" (dropdown menu with "I" selected). At the bottom, there are two orange buttons: "Sign Up" and "Reset".

Figure 1: User Registration

2. Login (Attribute Based Access)

Attribute-based access control is used in the suggested system's login process to make sure that user authentication is safe and flexible. When a person logs in, they give their passwords along with the set of traits they were given when they registered. The system checks these characteristics against access rules that have already been set up and are tied to encrypted data resources.



The login form features a large orange circular icon with a white padlock and the text "LOG IN" at the top. Below this, the form contains two input fields: "Email ID" (filled with "deepak@gmail.com") and "Password" (masked with dots). At the bottom, there is a single orange button labeled "LogIn".

Figure 2: User Login

Access is only given when the user's characteristics meet the policy requirements. Figure 2 depicts attribute-based secure user login authentication process. By enforcing access rights based on user characteristics, this method improves security and gets rid of the need for basic identity verification.

3. Key Generation: Setup

a. Cp-Abe Based Key Generation

The suggested system uses the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) method to make encryption keys that are linked to user characteristics. At first, the system goes through a setup process that creates the public parameters and a master secret key. These are important parts of any cryptography actions that follow.

Algorithm

Step-wise algorithm for CP-ABE Based Key Generation

Step 1: Global Setup

Let:

- G_0, G_1 be bilinear groups of prime order p
- $e: G_0 \times G_0 \rightarrow G_1$ be a bilinear map
- $g \in G_0$ be a generator

Select randomly:

- $\alpha, \beta \in \mathbb{Z}_p^*$

Compute:

- $h = g^\beta$
- $f = g^{\frac{1}{\beta}}$
- $e(g, g)^\alpha$

Public Key (PK):

$$PK = \left(G^0, g, h = g^\beta, f = g^{\frac{1}{\beta}}, e(g, g)^\alpha \right)$$

Master Key (MK):

$$MK = (\beta, g^\alpha)$$

Step 2: Key Generation for User with Attribute Set S

Let $S = \{\text{attributes possessed by user } u\}$

Choose:

- $r \in \mathbb{Z}_p$ (random value)

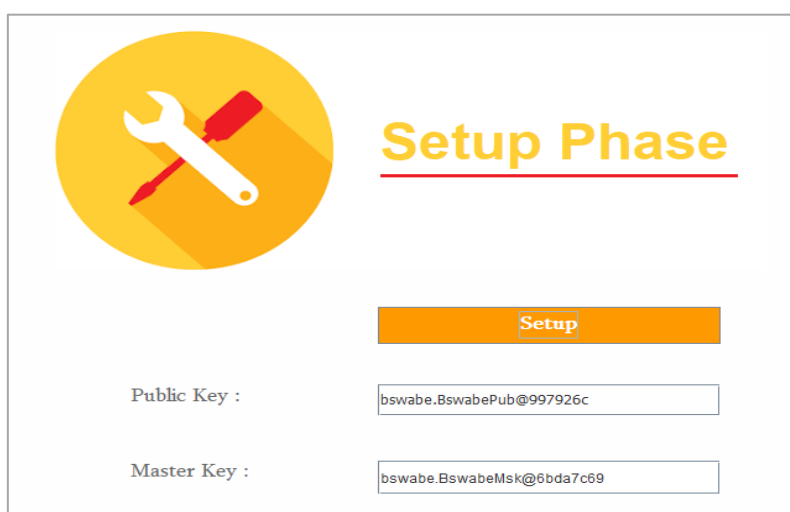
Compute main private key component:

$$D = g^{\frac{\alpha + r}{\beta}} = g^\alpha \cdot f^r$$

For each attribute $i \in S$:

- Choose random $r_i \in \mathbb{Z}_p$

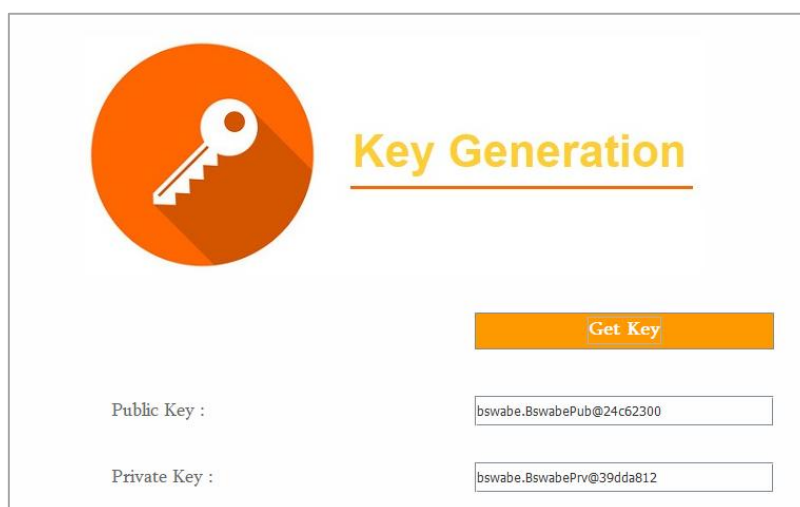
– Let $H(i) \in G_0$ be a collision – resistant hash function
 Compute:
 – $D_i = g^r \cdot H(i)^{r_i}$
 – $D'_i = g^{r_i}$
 Final User Private Key:
 $SK = (D = g^{\frac{\alpha+r}{\beta}}, \{D_i = g^r \cdot H(i)^{r_i}, D'_i = g^{r_i}\} \text{ for each } i \in S)$



The interface for the Setup Phase features a yellow circular icon with a white wrench and red pencil. The title "Setup Phase" is in yellow text with a red underline. An orange "Setup" button is at the top right. Below it, the "Public Key" field contains the text "bswabe.BswabePub@997926c". The "Master Key" field contains the text "bswabe.BswabeMsk@6bda7c69".

Figure 3: Key Generation Setup Phase

These factors are used to make secret keys for people based on the characteristics that have been given to them. This allows for fine-grained control of access. Figure 3 illustrates initial setup for attribute-based key generation process.



The interface for the Key Generation phase features an orange circular icon with a white key. The title "Key Generation" is in yellow text with an orange underline. An orange "Get Key" button is at the top right. Below it, the "Public Key" field contains the text "bswabe.BswabePub@24c62300". The "Private Key" field contains the text "bswabe.BswabePrv@39dda812".

Figure 4: Key Generation Phase


```
sn:I
cn:abc
uid:Computer
att :objectClass/inetOrgPerson objectClass:organizationalPerson sn:I cn:abc uid:Computer title:Teacher

sn:I
cn:abc
uid:Computer
<<<<<<<<<<>>>>>>>>>>>>>>>>>>
attrs : objectClass/inetOrgPerson
attrs : objectClass:organizationalPerson
attrs : sn:I
attrs : cn:abc
attrs : uid:Computer
attrs : title:Teacher
123862424      6546.0
attribute : Teacher
attribute : Computer
attribute : I
attribute : abc
10010100001010000001100000100010000100110010000000|
```

Pairing-based cryptography is used during setup to provide strong security promises while keeping the computing speed that is good for IoT-cloud settings.

a. File Browse

Open

Look In: input

- aaa.txt
- bbb.txt
- ccc.txt
- ddd.txt
- eee.txt
- test.txt

File Name: bbb.txt

Files of Type: All Files

Open Cancel

72

This feature gives authorised users an easy-to-use interface for viewing files that are saved locally or in the cloud. Figure 6 displays user interface for browsing files securely before encryption. Attribute-based encryption sets access control rules that are respected by the viewing process. This makes sure that users can only see files that they are allowed to access. By adding secure viewing, the system lowers the chance that files will be seen by people who shouldn't be able to and makes it easier to work with protected data. This method for controlled file selection sets the stage for later secure file actions, like encrypting and decrypting, which keep data safe and private throughout the user's routine.

b. File Encrypt

i. Lightweight AES Encryption

The suggested system's file encryption uses lightweight AES encryption to keep data safe with little extra work for computers, which is very important for IoT devices that don't have a lot of resources. The Advanced Encryption Standard (AES) is a symmetric key encryption method that is widely used and known for being strong and quick. Figure 7 illustrates lightweight AES encryption applied to selected files securely.

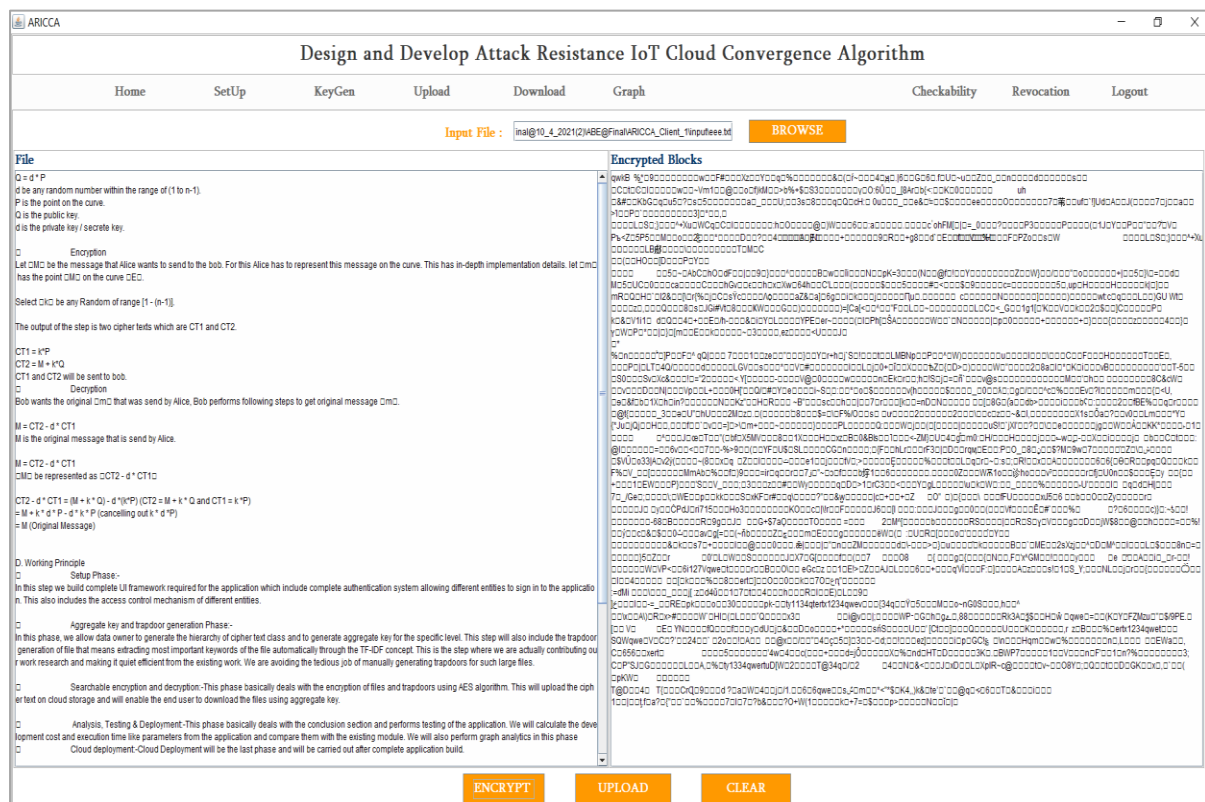


Figure 7: File Encryption

A lightweight version of AES is built into the system, which makes encryption and decoding quick while still providing strong security. This combined method works well with the attribute-based encryption scheme because it encrypts the file content quickly, and CP-ABE

keeps the encryption keys safe by controlling who can see them. Lightweight AES encryption uses little power and has a short working time, so it works well in IoT-cloud settings where speed and energy economy are important.

Algorithm for AES Encryption

Step 1: Key Expansion (Generate Round Keys)

Given: Cipher Key K (128 bits)

$\rightarrow W[0], W[1], \dots, W[43]$ (each $W[i]$ is 32 – bit word)

For $i = 4$ to 43:

if $i \bmod 4 = 0$:

$W[i] = W[i - 4] \oplus T(W[i - 1])$

where $T(x) = \text{SubWord}(\text{RotWord}(x)) \oplus \text{Rcon}\left[\frac{i}{4}\right]$

else:

$W[i] = W[i - 4] \oplus W[i - 1]$

Step 2: Initial Round (AddRoundKey)

Input: Plaintext Block P (4x4 matrix of bytes) \rightarrow State

Initial transformation:

$\text{State} = \text{State} \oplus \text{RoundKey}_0$

where $\text{RoundKey}^0 = [W[0], W[1], W[2], W[3]]$ reshaped to 4x4 matrix

Step 3: Nr-1 Main Rounds (Nr = 10 for AES-128)

Repeat for round = 1 to 9:

Each round includes 4 operations:

1. SubBytes:

For each byte b in State:

$b = \text{SBox}[b]$ (use non-linear substitution via Rijndael S-box)

2. ShiftRows:

Cyclically left-shift each row by its row index:

Row 0: no shift

Row 1: shift by 1

Row 2: shift by 2

Row 3: shift by 3

3. MixColumns:

For each column c in State:

$c' = M \times c$ (matrix multiplication in $GF(2^8)$)

$M =$

[02 03 01 01]

[01 02 03 01]

[01 01 02 03]

[03 01 01 02]

4. AddRoundKey:

$$State = State \oplus RoundKey_{round}$$

where $RoundKey_{round}$

$$= [W[4r], W[4r + 1], W[4r + 2], W[4r + 3]] \text{ reshaped to } 4 \times 4 \text{ matrix}$$

Step 4: Final Round (No MixColumns)

Repeat only 3 operations:

- SubBytes
- ShiftRows
- AddRoundKey using final round key

Step 5: Output Ciphertext

Output: Ciphertext C = State (after final round) serialized column-wise

c. Hash Generation

i. SHA 256

SHA-256 hash creation is a key part of making sure that data is correct in the IoT cloud convergence system. Safe Hash Algorithm 256-bit, or SHA-256, is a secure hash function that takes any raw data and returns a fixed-size 256-bit hash value. This hash is like a digital fingerprint for the file; it lets you know if it has been changed without your permission while it is being stored or sent. By making a SHA-256 hash of the protected file before uploading it, the system makes sure that files aren't changed by letting later steps check the security of the data. Changes, errors, or data loss are prevented by this process. SHA-256 is good for IoT settings that need solid but light security checks because it doesn't easily collide with other data and is fast to compute.

d. File Upload

The file upload feature in the IoT framework lets users safely send protected files from local devices to cloud storage. This process works closely with the system's security features to make sure that only files that are properly encrypted using attribute-based rules and come with integrity hashes are accepted. The upload module checks a user's rights based on their characteristics before sending a file. This stops people from uploading data without permission. The system also has efficient and reliable file methods to deal with the unpredictable nature of networks in IoT situations. The system protects data privacy, stops harmful injections, and makes sure that cloud storage only holds real, authorised data by using attribute-based access controls and integrity checking during file uploads.

5. File Download

a. Select File

During the download process, users can view and pick protected files that they are allowed to access based on their attribute details. This module works with the attribute-based access control system to limit the files that each user can see. It does this by making sure that only

files whose access rules match the user's characteristics are shown. By using attribute-based filtering, the system reduces attempts by people who aren't supposed to be there to get in and makes it easier to use by showing only the relevant data. The easy-to-use file selection layout makes it easier for users to find the files they need quickly and safely.

b. Verify File (Auditing)

File checking through auditing is done by the system before the download starts to make sure the desired data is correct and real. This process checks the SHA-256 hash of the protected file that was saved against the hash that was first calculated to find any changes, errors, or hacking that were not authorised. The monitoring tool also checks that the user's credentials match the file's access policy, which stops anyone from downloading without permission. This two-layer verification makes data safer by making sure that only authorised users can view files that haven't been changed.

c. File Download

As soon as the file selection and verification steps are finished successfully, the system sends the protected file safely to the authorised user. Secure communication methods are used during the download process to keep data in transit from being viewed or changed by attackers. Figure 8 shows secure file download process with attribute-based access control.

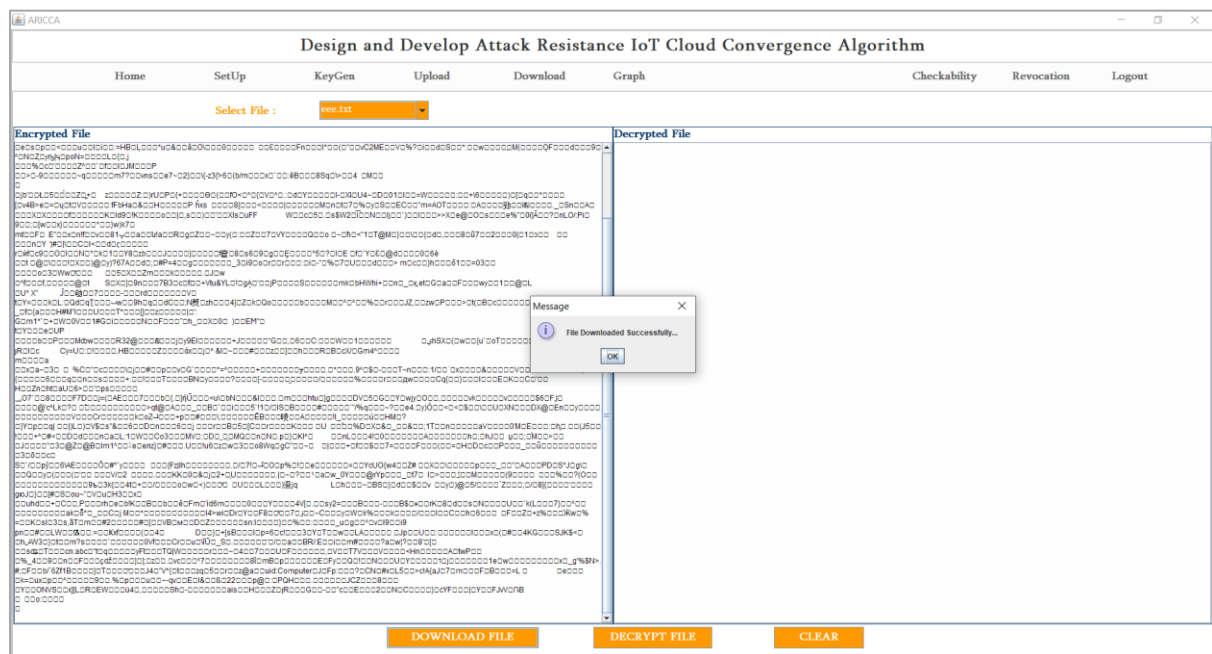


Figure 8: File Download

The system might also be able to resume files so that it can handle the unstable networks that are common in IoT settings. After downloading, the user can use their attribute-based private keys to get to the original raw data during the decoding process.

d. Decrypt File

The suggested system uses the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) method to decrypt files. Figure 9 depicts policy verification followed by secure attribute-based file decryption. This makes sure that only users whose characteristics match the inserted access policy can decrypt files correctly.

```

policy : sn:I cn:abc uid:Computer 3of3
m = {x=72552353197094483650043878604962637463553787308733772706067946868
7654355668612928360459326059637632624,y=83470278535217737661290098017405
0888230751789314638118031112142603619999451741541287883433419923110925}
connected
isflag : false
outsourcedDecryptionMemory : 4860784.0
outsourced

```

Figure 9: Policy Matching and File Decryption

The user uses their attribute-based private key to start decryption as soon as they receive the encrypted file. Before showing the original raw data, this cryptography process checks that the user's properties meet the access policy that was set during encryption. Figure 10 illustrates secure decryption of files using attribute-based keys.

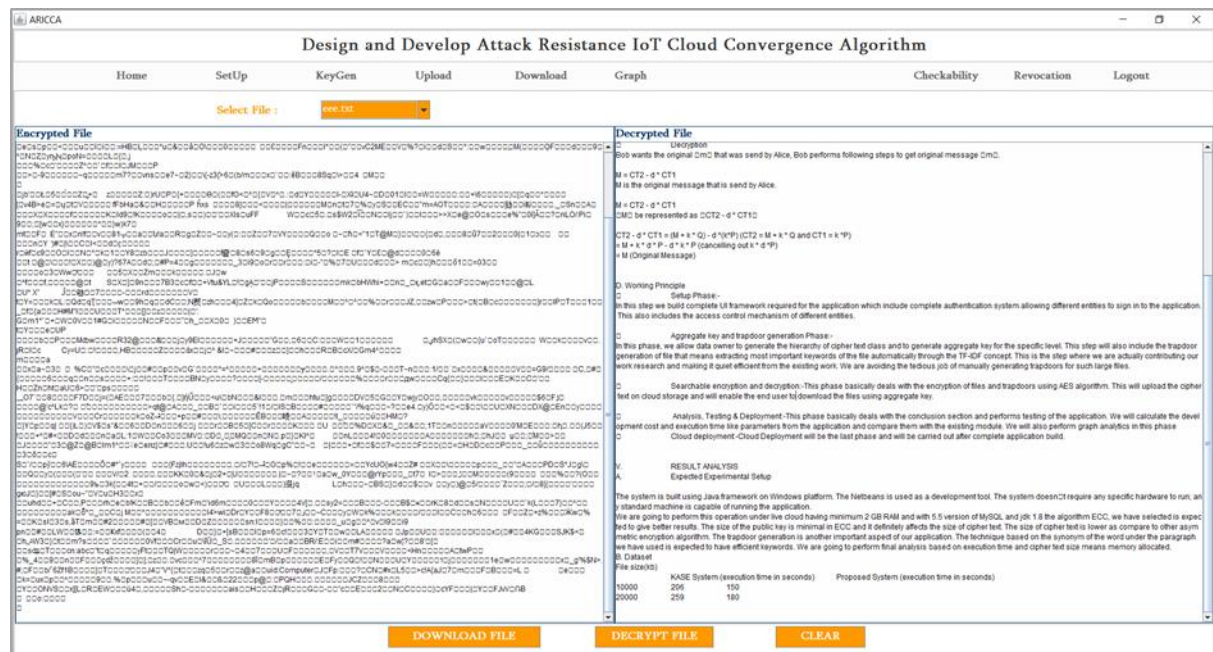


Figure 10: File Decryption

During the decoding process, pairing-based encryption operations may be used. These operations offer strong security while still being efficient. This attribute-driven decryption applies fine-grained access control, so even if someone gets the protected file, they can't get to sensitive information without permission.

6. TPA / Attack Module

a. Perform Checkability (Attribute Verification)

The Perform Checkability module is an important defence tool because it checks that user traits are correct against set access rules. Figure 11 shows attribute verification ensuring correct access rights granted.



Figure 11: Checkability (Correct Attributes) – Anagram

This attribute checking makes sure that only people with true and authorised attributes can access sensitive data or do sensitive actions. As a part of the Third Party Auditor (TPA) tool, it stops efforts at unauthorised entry by finding attribute errors or fakes. Figure 12 illustrates attribute verification failure due to incorrect user attributes.

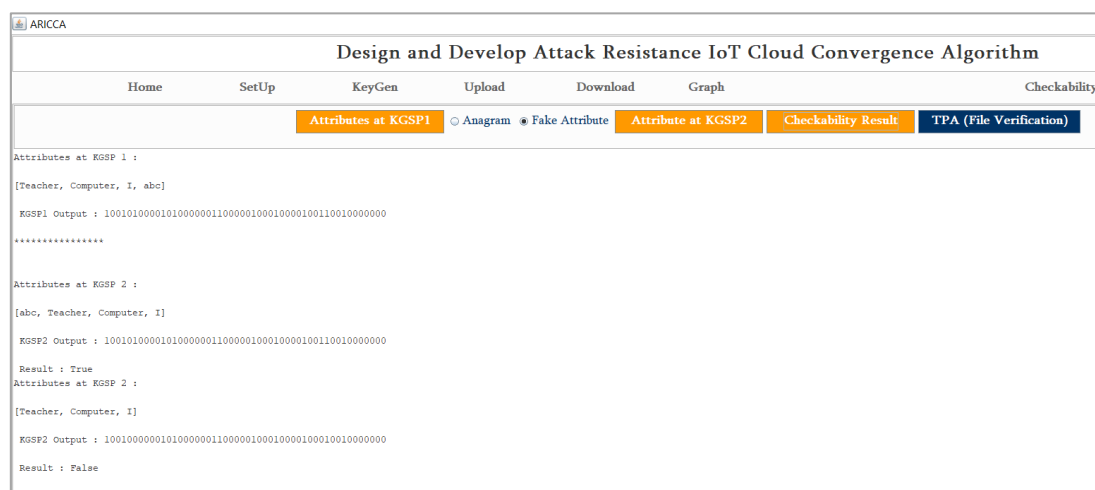


Figure 12: Checkability (Wrong Attributes)

This method keeps the system safe in changing IoT-cloud settings by constantly checking the validity of attributes during access requests. This makes the system less vulnerable to attacks like attribute faking or collusion.

b. File Verification Using HASHING

Verifying files with hashing is an important part of making sure that data in the IoT-cloud system is correct and real. Before letting you view or download the protected file, the system makes an encryption hash of it (e.g., SHA-256) and checks it against the hash value that has been saved. Figure 13 depicts secure outsourced file decryption with integrity verification. Any difference could mean that something has been changed or corrupted.



Figure 13: Outsource Decryption and Verification of File

This approval method keeps the file from being changed without permission, so it stays the same from uploading to retrieving it. By using hashing, the system makes checking quick and safe without showing the actual file content. This protects privacy and keeps data saved in the cloud trustworthy.

c. User Revocation

A key part of the security framework is user removal, which lets the system take away users' access rights when their information changes or their passwords are stolen. The suggested method lets the data owner or an authorised authority handle removal. Figure 14 shows data owner revoking user access and permissions securely.



Figure 14: User Revocation by Data Owner

They would change the attribute-based access rules and make the banned user's private keys useless. This stops the person whose permission was removed from decrypting any data, whether it's already there or not. Figure 15 illustrates process of removing revoked user's decryption privileges. This keeps private data safe.

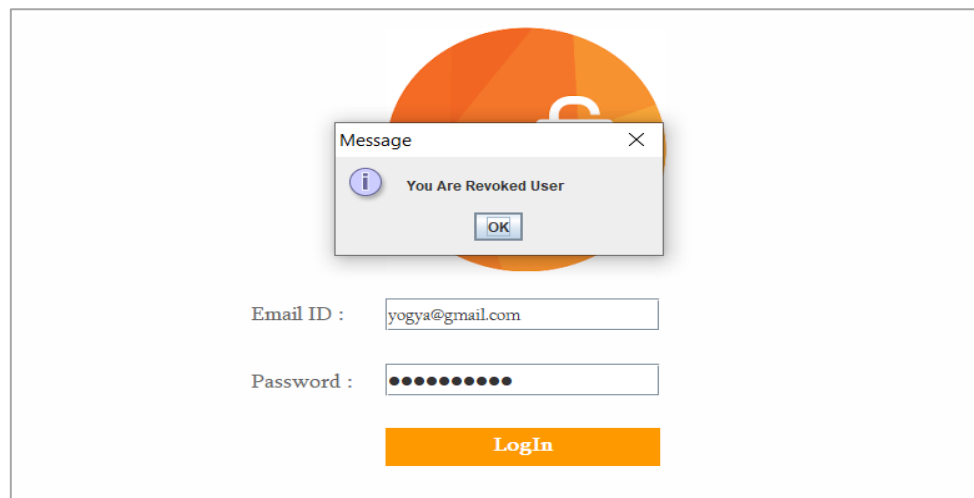


Figure 15: User Revocation

Revocation methods that work well keep the system safe without slowing it down too much or blocking legal users' access, which is very important in IoT-cloud settings where user jobs are always changing.

IV. Performance Analysis

a. File Insource Download and Decryption

Performance study of file insource download and decoding checks how well it works to work with data in the user's own space. The results show that insource decryption is safe, but it needs a lot of computing power, which could slow down IoT devices that don't have a lot of power. To combine security with usefulness in real-world IoT-cloud situations, this method needs to be optimised. The File Insource Decryption Time Comparison Graph in Figure 16 shows the difference in decryption time (in milliseconds) between the current system and the suggested system.

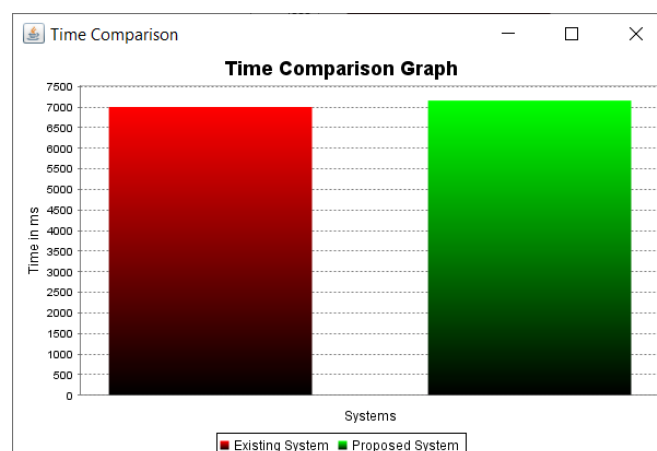


Figure 16: File Insource Decryption Time Comparison Graph

The File Insource Decryption Memory Comparison Graph (Figure 17) shows how much memory the current and suggested methods use in bytes. The suggested system uses about 60,500,000 bytes, which is a little less than the current system's 61,000,000 bytes.

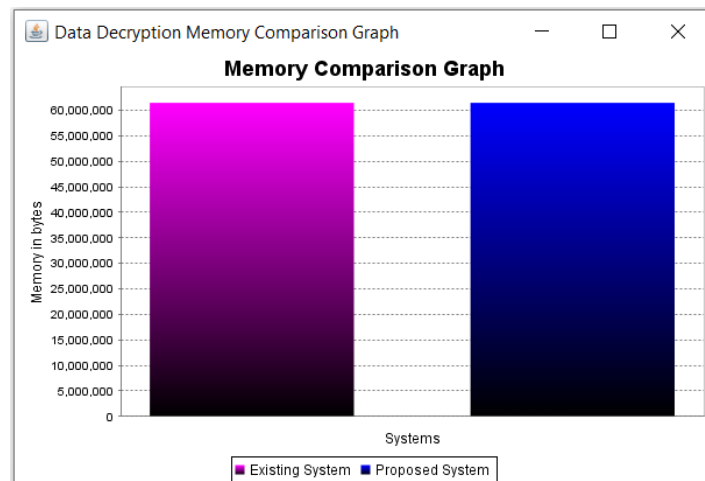


Figure 17: File Insource Decryption Memory Comparison Graph

This lower memory use shows that the suggested system is more efficient, even though it has better security features. The algorithm's smaller memory size makes it better for IoT settings with limited resources. It strikes a mix between speed and security without adding a lot of extra memory when decrypting files.

b. File Outsource Download and Decryption

The File Outsource decoding Time Comparison Graph is shown in Figure 18. It shows the decoding times in milliseconds for both the current and suggested methods. It takes about 7000 milliseconds for the current system to fully decrypt, but only 6400 milliseconds for the suggested system, which makes it faster. This big drop of about 600 ms shows how useful the suggested method is for situations where decoding is outsourced.

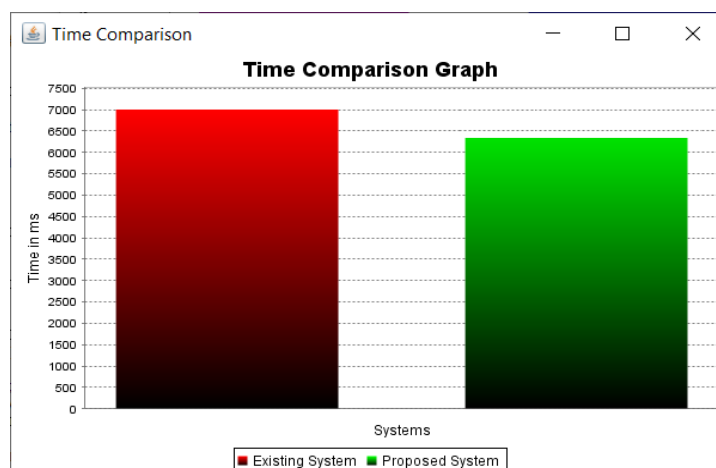


Figure 18: File Outsource Decryption Time Comparison Graph

Figure 19 shows the File Outsource Decryption Memory Comparison Graph, which shows how much memory the current system and the suggested system use.

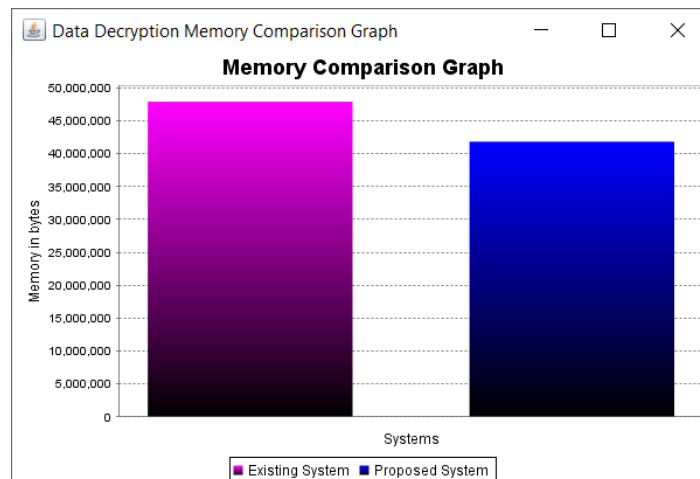


Figure 19: File Outsource Decryption Memory Comparison Graph

The suggested system uses only 42,000,000 bytes of memory, a huge reduction from the 47,500,000 bytes used by the current system. This decrease of about 5,500,000 bytes shows that the suggested method uses memory more efficiently during external decoding processes.

c. Comparative Analysis

i. Time Comparison Graph

Table 2 compares the decryption times for files of different sizes, showing that the times get longer as the file sizes get bigger. This process takes 710 ms for a file that is 2890 KB in size. From 3120 KB to 6075 KB, this time goes up to 845 ms, 1182 ms, 1300 ms, and 1520 ms, with the biggest file size of 6075 KB.

Table 2: Decryption Time Comparison for various File sizes

File Size (KB)	Time (ms)
2890	710
3120	845
4858	1182
5230	1300
6075	1520

The data consistently shows an upward trend, which means that decrypting bigger files takes a longer time overall. This trend shows that the system can be scaled up because the time it takes to decode a file increases regularly with its size. Figure 20 compares decryption times across different dataset sizes efficiently. This is useful for planning how to use resources in IoT-cloud settings.

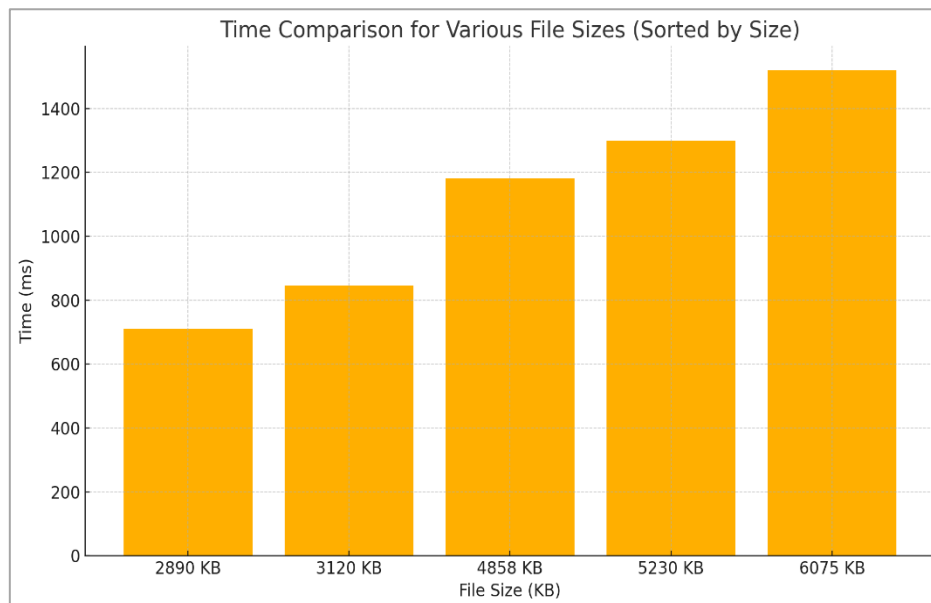


Figure 20: Decryption Time Comparison Graph for various Dataset sizes

Table 3 shows how much memory is used for decryption for files of different sizes, showing that the size of the file makes more memory needed. The amount of memory used for a file that is 2890 KB is about 4,999,060 bytes. The next largest file size is 10,508,404 bytes, which is for a file that is 6075 KB in size.

Table 3: Decryption Memory Comparison for various File sizes

File Size (KB)	Memory Usage (Bytes)
2890	4999060
3120	5396909
4858	8403264
5230	9046742
6075	10508404

The next largest file size is 8,403,264 bytes, which is for a file that is 4858 KB in size. There is a clear link between file size and the amount of memory needed for decryption, as shown by the data. Figure 21 compares memory usage during decryption for different dataset sizes.

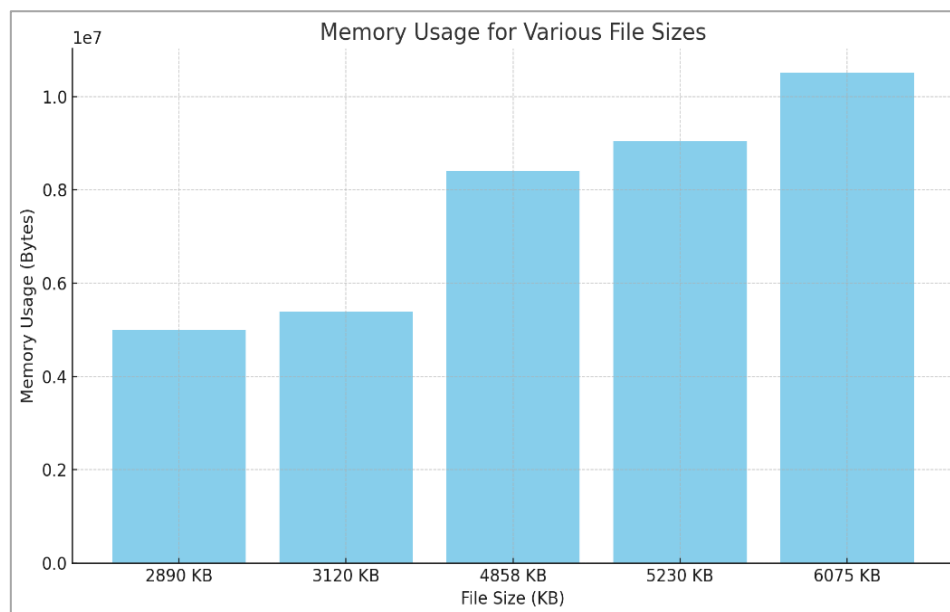


Figure 21: Decryption Memory Comparison Graph for various Dataset sizes

IoT-cloud settings need to be able to control memory well, and these results show that the system can handle rising resource needs as file sizes increase in a predictable way.

V. Conclusion

This study described how to create an IoT cloud convergence method that is hard to attack. It uses Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to improve data security, fine-grained access control, and system resilience. The suggested framework solves some of the most important problems that come up when connecting IoT devices to the cloud. These problems include safe key management, dynamic user attribute verification, and efficient encryption and decoding processes that work well in IoT settings with limited resources. The system strikes a good mix between security and speed by using CP-ABE along with lightweight AES encryption and SHA-256 hashing. This keeps private data safe from people who shouldn't have access to it and makes sure that the data is correct. The attribute-based registration and login tools let you handle users in a way that is both flexible and scalable. They do this by enforcing access rules precisely based on user characteristics instead of static names. This adaptability helps IoT-cloud environments that are always changing, where user jobs and rights are always changing. The Third Party Auditor (TPA) feature also improves trustworthiness by making sure that data is real and allowing safe user removal, which stops users who have been banned from getting protected resources. The suggested method takes less time and uses less memory than current systems, even though it has more security features. This was shown by performance study results.

References

- [1] Li, J.; Wang, Y.; Zhang, Y.; Han, J. Full Verifiability for Outsourced Decryption in Attribute Based Encryption. *IEEE Trans. Serv. Comput.* 2020, 13, 478–487.
- [2] Zhang, R.; Li, J.; Lu, Y.; Han, J.; Zhang, Y. Key Escrow-free Attribute Based Encryption with User Revocation. *Inf. Sci.* 2022, 600, 59–72.
- [3] Chen, N.; Li, J.; Zhang, Y.; Guo, Y. Efficient CP-ABE Scheme with Shared Decryption in Cloud Storage. *IEEE Trans. Comput.* 2022, 71, 175–184.
- [4] Li, J.; Zhang, E.; Han, J.; Zhang, Y.; Shen, J. PH-MG-ABE: A Flexible Policy-Hidden Multi-Group Attribute-Based Encryption Scheme for Secure Cloud Storage. *IEEE Internet Things J.* 2024.
- [5] Chen, S.; Li, J.; Zhang, Y.; Han, J. Efficient Revocable Attribute-based Encryption with Verifiable Data Integrity. *IEEE Internet Things J.* 2024, 11, 10441–10451.
- [6] Sivasankari, N.; Kamalakkannan, S. Detection and prevention of man-in-the-middle attack in iot network using regression modeling. *Adv. Eng. Softw.* 2022, 169, 103126.
- [7] Chaudhary, S.; Mishra, P.K. DDoS attacks in Industrial IoT: A survey. *Comput. Netw.* 2023, 236, 110015.
- [8] Yang, W.; Wang, S.; Yin, X.; Wang, X.; Hu, J. A review on security issues and solutions of the Internet of Drones. *IEEE Open J. Comput. Soc.* 2022, 3, 96–110.
- [9] Harbi, Y.; Aliouat, Z.; Harous, S.; Bentaleb, A.; Refoufi, A. A review of security in internet of things. *Wirel. Pers. Commun.* 2019, 108, 325–344.
- [10] Yousefnezhad, N.; Malhi, A.; Främling, K. Security in product lifecycle of IoT devices: A survey. *J. Netw. Comput. Appl.* 2020, 171, 102779.
- [11] Ling, L.; Yelland, N.; Hatzigianni, M.; Dickson-Deane, C. The use of Internet of Things devices in early childhood education: A systematic review. *Educ. Inf. Technol.* 2022, 27, 6333–6352.
- [12] Shah, K.; Sheth, C.; Doshi, N. A survey on iot-based smart cars, their functionalities and challenges. *Procedia Comput. Sci.* 2022, 210, 295–300.
- [13] Ootom, A.F.; Eleisah, W.; Abdallah, E.E. Deep learning for accurate detection of brute force attacks on IOT Networks. *Procedia Comput. Sci.* 2023, 220, 291–298.
- [14] Sanlı, M. Detection and Mitigation of Denial of Service Attacks in Internet of Things Networks. *Arab. J. Sci. Eng.* 2024, 49, 12629–12639.
- [15] Aziz Al Kabir, M.; Elmedany, W.; Sharif, M.S. Securing IoT devices against emerging security threats: Challenges and mitigation techniques. *J. Cyber Secur. Technol.* 2023, 7, 199–223.
- [16] Ren, Y.; Peng, H.; Li, L.; Xue, X.; Lan, Y.; Yang, Y. A voice spoofing detection framework for IoT systems with feature pyramid and online knowledge distillation. *J. Syst. Archit.* 2023, 143, 102981.
- [17] Qasem, M.A.; Thabit, F.; Can, O.; Naji, E.; Alkhzaimi, H.A.; Patil, P.R.; Thorat, S. Cryptography algorithms for improving the security of cloud-based internet of things. *Secur. Priv.* 2024, 7, e378.

- [18] Fathalizadeh, A.; Moghtadaiee, V.; Alishahi, M. On the privacy protection of indoor location dataset using anonymization. *Comput. Secur.* 2022, 117, 102665.
- [19] Arul, R.; Alroobaea, R.; Tariq, U.; Almulihi, A.H.; Alharithi, F.S.; Shoaib, U. IoT-enabled healthcare systems using block chain-dependent adaptable services. *Pers. Ubiquitous Comput.* 2024, 28, 43–57.