**JOURNAL OF ALGEBRAIC STATISTICS**
Volume 13, No. 3, 2022, p. 3981-3995
https://publishoa.com
ISSN: 1309-3452

# An effective Cloud Access Control Mechanism based on Attribute-based Key Derivation Technique

Suresha D[1] and Dr. K Karisiddappa[2]

[1]Research Scholar, VTU, Belagavi, Associate Professor, Dr. AIT, Bangalore

[2]Professor, Dayanand Sagar Academy of Technology and Management, Bangalore

**Received** 2022 March 15; **Revised** 2022 April 20; **Accepted** 2022 May 10.

**Abstract**

Cloud computing provides a platform for users to store and manage data. The cloud stores massive amounts of data and may be accessed by various people. Though the cloud provides data protection, the owners of vital data are concerned about the security provided by the cloud. The data owner must secure the data that will be stored in the cloud, and there is also concern about data access from the cloud. Traditional cryptographic approaches, such as symmetric and asymmetric encryption and decoding, are unsuitable for cloud data privacy and access management. To address this, attribute-based encryption is an appropriate approach for better privacy and an improved access control mechanism. To improve cloud data access, an attribute-based key generation and encryption system has been implemented. This article also allows cloud users to access only the data they need. The experimental results show that the proposed approach reduces the processing cost for key derivation. The solution ensures that registered cloud users have access to their valuable data in the cloud.

**Keywords**: Access control; cloud storage; confidentiality; data origin authentication; key derivation

## 1. Introduction

Cloud computing is a technology that stores enormous volumes of data on a network of cloud computers located in multiple geographical locations. Because massive amounts of data are stored in cloud servers, security concerns such as confidentiality, authenticity, and privacy preservation are critical. Cryptographic algorithms are the only ones capable of meeting these security requirements. Cryptographic techniques are critical in assuring the security of cloud-stored data. Because the cloud allows millions of users to access data, all users must follow security rules when doing so.

The intended cloud user will be granted data access permissions, and before accessing the data, the cloud user must first register with the cloud service provider (CSP). The cloud service provider (CSP) cannot always be trusted to share data with the intended cloud user. As a result, an unauthorized party may get access to or divulge sensitive data to the cloud service provider. Attribute-based encryption is a contemporary trend in cloud computing in which data relevant to the cloud user is encrypted using a key created from the cloud user's attribute and transmitted via a secure communication channel with such a concerned user.

Khalid Albulayhi et al. [1] did an analytical study of fine-grained access control techniques in cloud computing. Attribute-based access control (ABAC) is a significant algorithm that considers encryption, transparency, categorization, and authorization. Another access control approach that separates data into two pieces is Key-Policy Attribute-Based Encryption (KP-ABE). The information body is encrypted using a key derived from the header component. They compared sophisticated access control mechanisms based on characteristics such as processing overhead, backward security, fine-grained access control, and architecture.

Neenu Garg et al. [2] presented a data integrity protocol for cloud computing. The protocol's major goal is to reduce the complexity of the cloud user's computation based on bilinear pairings. The experimental results overcome the computational Diffie Hellman Problem (CDHP) constraints in a Random Oracle Model (ROM). The findings were compared to current protocols with restricted client resources, which decreases the cost of metadata calculation on the client during the system setup phase of the auditing protocol.

Smarajit Ghosh et al. [3] suggested weighted attribute based encryption based on the Blowfish algorithm for cloud computing security and data sharing. Each characteristic is given a weight, and access control mechanisms are implemented to encrypt the data. Clients, on the other hand, can retrieve the relevant data file based on its weight to reduce computational burden. In terms of data secrecy, data cooperation, and partial decryption, the prototype is built and compared to the hybrid attribute-based encryption (HABE) method. Verification is performed on both encrypted and decrypted data. When the cloud delivers erroneous findings, the cloud user can quickly detect them by running the verification process. When compared to ABE on the user side, this approach has a lower computational cost.

Qiuting Tian et al. [3] developed a hierarchical authority based weighted attribute encryption method in cloud computing. In order to increase data security, the technique is executed using online or offline encryption. In order to provide fine-grained access control, weights are assigned to characteristics in the encryption scheme, and attribute private keys are distributed to users at different levels. Based on the attribute weights, the system security is enhanced at each user level. Ajay Kumar Dubey et al. [4] suggested attribute-based credit calculation as a data storage access control method in cloud data centres. A brief overview of the access control mechanisms is provided, as well as their drawbacks. The suggested method's essential features, such as encouraging reputable users, discouraging harmful users, and credit computation, were explored. Cloudsim, a network simulator, is shown in action for assessing experimental findings.

Jialu Hao et al. [5] suggested a fine-grained attribute concealing policy for cloud-based internet of things (IoT). The method may conceal attribute information while also supporting the access policy to the fullest degree possible. To restrict unwanted access to the ciphertext, a fuzzy attribute technique is devised for ciphertext decryption. The experimental results preserve privacy while requiring less compute and storage. Shengmin Xu et al. [6] proposed fine-grained bilateral access control for secure cloud-fog computing. They proposed a new concept known as cloud-fog computing, which would allow them to provide a variety of on-demand services over a network. The identification and retrieval of useful data from a large volume of ciphertext data without the use of expensive decryption mechanisms remains a difficult problem. They created a system that provides confidentiality as well as data origin identification by utilizing a new cryptographic technique known as matchmaking attribute-based encryption (MABE).

Sana Belguith et al. [7] proposed a privacy-preserving attribute-based framework for cloud access control. To distribute keys among cloud users, existing cryptographic techniques for access control impose a high computational cost on the data owner's site. They created a system that combines attribute-based encryption and attribute-based signature mechanisms to allow for the secure sharing of outsourced data via the public cloud. Mahender Kumar and Satish Chand [8] proposed a secure key issuing identity-based encryption in a cloud environment. Because of the high cost of storage, traditional public key cryptosystems are impractical in real-time environments. To overcome the limitations of traditional public key cryptography, the author proposed a mechanism for generating the user's private key that eliminates the need for a key generation centre (KGC). To secure communication over a public channel, the ECC-based blind technique is used.

Shengmin Xu et al. [9] proposed fine-grained access control for dynamic groups in a cloud environment. Cloud computing is a new trend in which users can store data and access scalable on-demand services. At the same time, cloud computing introduces numerous security issues because cloud service providers (CSPs) do not share the same trusted domain as users. Yi Liu et al. [10] proposed a secure and fine-grained access control mechanism for storing electronic health care records in mobile cloud computing. For access control of e-health care records, the bilinear Diffie-hellman exponent assumption is used. The simulation results have been generated, and the model has been determined to be suitable for mobile cloud computing.

Fangbo Cal et al. [11] conducted a survey on various access control techniques to protect information and system resources and to limit unauthorised users in a cloud environment. They summarise the benefits and drawbacks of various access control mechanisms and suggest some future research directions. Shivanna K et al. [12] proposed a double encryption method for cloud computing privacy preservation. They proposed that using double encryption is essential for improving the privacy of data stored on cloud servers. They proposed two different algorithms, one for storing data and the other for accessing data from the cloud server.

Rashad Elhabob et al. [13] proposed identity-based encryption with an authorized equivalence test for cloud-aided IoT. They built a system to store the data collected by the sensors in the cloud. Before storing data in the cloud, that data should be encrypted. They proposed an identity-based encryption with authorized equivalence test (IBE-AET) to test the equivalence of two messages encrypted with the same or different identities. They put the system through its paces with simulated results and theoretical analysis. Kwangsu Lee [14] proposed a revocable identity-based encryption method based on the subset difference method rather than the complete subtree (CS) method. They created the system by combining identity-based and hierarchical identity-based encryption.

Hongbing Cheng et al. [15] proposed a method for accountable privacy preservation based on identity-based encryption. The system architecture for accountable privacy preservation, as well as a detailed security analysis, is presented. The system is designed to defend against various types of attacks, and the simulation results in increased efficiency. Chandrashekhar Meshram et al. [16] proposed an identity-based encryption technique based on fuzzy technique for data sharing in cloud environment. To reduce key disclosure, a technique is implemented that protects against a chosen ciphertext attack and a chosen sub-tree attack. This technique has the potential to overcome the limitations of currently available methods in terms of security and public key length.

For data sharing in a cloud setting, Chandrasekhar Meshram et al. [17] presented an identity-based encryption solution based on a fuzzy technique. To reduce key leakage, a strategy that guards against a chosen cipher text attack and a chosen sub-tree attack is implemented. This methodology has the potential to overcome the security and public key length restrictions of currently known solutions. Hu Deng et al. [18] [19] [20] presented Identity-Based Encryption for cloud data security with flexible data sharing among users. They presented two types of algorithms: identity-based encryption (IBE) and identity-based broadcast encryption (IBE) (IBBE). The primary goal of this study is to address the critical issue of identity revocation and offer a revocable IBE scheme in the server-aided setting. Nishat Farjana and colleagues [21] [22] [23] [24] safe identity-based data sharing approach for cloud computing social networks This scheme's major goal is to allow secure data sharing over authorized users. This method is resistant to keyword guessing attacks and has a lower computational cost.

Sharma et al. [25] [26] [27] presented a Blockchain-based architecture that employs identity-based encryption. Blockchain and the Internet of Things (IoT) are two dominating areas of information technology in today's globe. These technologies are prevalent in the supply chain, logistics, and automotive industries. They advocated using block chain technology to increase data sharing and security. The secure sharing of health records can improve the treatment process by improving diagnosis accuracy, security, and privacy. The architecture they created is entirely based on the Identity-Based Encryption (IBE) algorithm.

## 2. Design considerations

The proposed scheme is intended to ensure flexible access control operation while minimizing computational and storage overheads. The scheme involving notations and their meanings as illustrated in Table 1.

**Table 1: Notations and meaning**

| Notations | Descriptions |
|---|---|
| MasKey1 | master key$^1$ |
| Df | data file |
| Ed f | encrypted data file |
| TPA | third party auditor |
| Cu | cloud user |
| E | Encryption |
| D | Decryption |

| | |
|---|---|
| Acu | attributes of cloud user |
| EK | extracted key from MasKey1 |
| $SecKey^s$ | one time secret key set |
| $H^s$ | set of hash tag |
| $SHA_1$ | Secure hash algorithm[1] |
| Dbj | number of data blocks of Df |
| E(Dbj) | encrypted Dbj |
| D(Dbj) | decrypted Dbj |
| CSP | cloud service provider |

## 2.1 Design goals

The proposed methodology is implemented with a following design goals:

1. The solution is intended to provide fine-grained access control for all registered cloud users, providing for greater data access flexibility.
2. To reduce computational overhead, the system is designed so that a number of secret keys can be created from a single master key.
3. In reality, key robustness is achieved by extracting random characters from the master key and concatenating these characters with a unique cloud user attribute.
4. The key resiliency is obtained by randomly separating 128 bits from 160 bits.

## 3. Proposed methodology

From the point of view of the data owner site the following points are considered:

1. The data owner derives the 128-bit master key[1] (MasKey1) using the usual Blowfish algorithm.
2. Encrypts the full data file with master key[1] and sends the encrypted data file to cloud storage.
3. Through a secure connection, the data owner transfers the master key[1] to the trusted third party (TPA).

From the point of view of the trusted third party (TPA), the following points are considered:

1. Using the master key1 given by the data owner, the trusted third party (TPA) decrypts the full data file. TPA computes random characters from the master key[1] and keep recording these characters for future key generation.
2. TPA computes the secret keys related cloud users using randomly extracted characters of the master \$key^1\$ and selecting suitable cloud user attributes.
3. TPA converts entire data file into number of blocks and encrypts requested blocks of the cloud user using suitable secret keys.
4. Whenever the cloud user requests the data, the TPA sends the encrypted data to the intended cloud user along with a suitable secret key.

From the cloud service provider (CSP) and cloud user point of view, the following points are considered:

1. The cloud service provider (CSP) allows the cloud user for registration to access the data from the cloud storage.
2. After the successful registration, the cloud user can access the data from the cloud by receiving the suitable secret keys from the TPA.

The purpose of **Algorithm 1** is to compute the master key1 (MasKey1) at the data owner's location. The proposed approach creates a 128-bit key using the standard Blowfish algorithm. Depending on how the key is utilized, it can be converted into a number, or bytes, or text. In this study, the encryption created by the Blowfish method is translated into string characters.

```
Algorithm 1   Deriving Master key$^1$ (MasKey$^1$) at the data owner site
Input:        Key generator algorithm (Blowfish)
Output:       Master key$^1$
1:            generate key using Blowfish algorithm
2:            initialize key with 128 bits
3:            initialize master key$^1$ (MasKey$^1$) $\leftarrow$ key
4:            convert master key$^1$ into a string
5:            record master key$^1$ for further processing
```

Outsourced data secrecy is a major issue in cloud computing, and **Algorithm 2** is designed to address it. The original data file ($D_f$) containing a number of blocks belonging to the concerned data owner is initially encrypted at the data owner site using master key1 (MasKey1). The encrypted data file ($Ed_f$) is transmitted to the cloud for processing. By supplying the MasKey1, the $Ed_f$ accessibility credentials are granted to a third party auditor (TPA) via a secure connection.

```
Algorithm 2   Encryption of data file at data owner site
Input:        $D_f$ , MasKey$^1$
Output:       $E_{d f}$
1:            receives $D_f$ from Algorithm 1
2:            compute $E_{d f}$ such that $E_{d f} \leftarrow E(MasKey^1[D_f])$
3:            store $E_{d f}$ to the cloud
4:            sends MasKey$^1$ to the TPA through secure channel
```

**Algorithm 3** is designed for reading encrypted data files ($Ed_f$) at a third-party auditor's location (TPA). The TPA obtains the MasKey$^1$ from the chosen data owner through a secure connection and decrypts the data file for further processing. When the targeted cloud user requests the data blocks, the data file is encrypted and divided into a number of blocks.

```
Algorithm 3   Decryption of data file at TPA site
Input:        Ed f , MasKey1
Output:       Df
1:            receives MasKey1 from the data owner
              through secure channel
2:            compute Df such that
```

```
        Df ←D(MasKey1[Ed f ])
3:      record Df for further processing
```

**Algorithm 4** shows that data access is only available if the cloud user registers with the cloud platform and submits credentials. Cu=Cu$_1$, Cu$_2$, Cu$_3$...Cu$_n$ must register with a cloud service provider (CSP) with at least three acceptable attributes, such as Acu= Acu1, Acu2, Acu3. The attributes of any eligible cloud user are recorded for future review.

```
Algorithm 4   Registration of cloud user at CSP site
Input:        $C_u=C_{u1}, C_{u2}, C_{u3}\ldots C_{un}$,
              $A_{cu}= A_{cu1}, A_{cu2}, A_{cu3}$
Output:       Success or Fail
1:            $C_u$ request for registration
2:            for $C_u$ from 1 to n do
                  $entry_1 \leftarrow A_{cu1}$
                  $entry_2 \leftarrow A_{cu2}$
                  $entry_3 \leftarrow A_{cu3}$
              end for
3:            record $A_{cu}$ for further processing
```

**Algorithm 5** is being developed at the TPA site for character extraction from the master key. An alternative method can be used to retrieve the key from the master key. We must first determine the length of the master key, which is the number of characters in the master key, before we can access the characters in the master key. The characters are extracted at random with a preset length, and the number of times the characters are extracted is purely governed by the number of cloud users. It is critical to set the key size to 256 bits in order to improve key randomness. Each random character is recorded in order to compute the number of secret keys.

```
Algorithm 5   Key extraction at TPA site
Input:        MasKey1
Output:       EK=EK1, EK2, EK3....EKn
1:            find length of MasKey1 such that
              len=lentgh(MasKey1)
2:            retrieve random characters from MasKey1
              where MasKey1=16, 32, 64....n characters
              for 1 to nth do
                  EK1 ← len/2 (MasKey1)
                  EK2 ← len/2 (MasKey1)
                  EK3 ← len/2 (MasKey1)
                  ...................
                  EKn ← len/2 (MasKey1)
              end for
```

```
3:          record EK1, EK2, EK3...EKn for further processing
```

**Algorithm 6** is built in such a way that an XOR operation is conducted between the extracted characters $E_K$ and the cloud user's principal attribute. It accepts input as a key attribute of any cloud user and generates output in the form of XOR blocks. Each cloud user attribute $A_{cu1}$ is XOR'd with a separate extracted character block $E_K$. If the lengths of $A_{cu1}$ and $E_K$ differ, append 0's to either $A_{cu1}$ or $E_K$. Because the XOR operation is conducted with equal inputs of identical size.

Following the XOR operations, outputs of the type $X_{R1}$, $X_{R2}$, $X_{R3,\ldots}$ $X_{Rn,}$ are created, and these output blocks are recorded for further processing.

```
Algorithm 6   XOR operation between EK and Acu1
Input:        Acu1, EK=EK1, EK2, EK3....EKn
Output:       XR=XR1, XR2, XR3....XRn
1:            receive Acu1 of cloud user
2:            receive EK from Algorithm 5
3:            for Cu from 1 to n do
                  derive XR1 ← (EK1 ⊕ Acu1)
                  derive XR2 ← (EK2 ⊕ Acu1)
                  derive XR3 ← (EK3 ⊕ Acu1)
                  ……..
                  derive XRn ← (EKn ⊕ Acu1)
              end for
4:            record XR1, XR2, XR3....XRn
               for further processing
```

**Algorithm 7** in this study is intended to improve the robustness of the key derivation. The outputs of Algorithm 6 are used as input for this stage, and they are processed using a circular left shift operation on $X_R$ blocks. During this stage of research, each individual block, $X_{R1}$, $X_{R2}$, $X_{R3,\ldots}$ $X_{Rn}$, is subjected to cyclic left shifting. The length of the shifting is determined by the size of the $X_R$ blocks. The final shifting blocks, $CLS_1$, $CLS_2$, $CLS_3$,…CLSn, are preserved for additional processing.

```
Algorithm 7   Circular left shifting on XR
Input:        XR=XR1, XR2, XR3....XRn
Output:       CLS=CLS1, CLS2, CLS3....CLSn
1:            receives XR=XR1, XR2, XR3....XRn from Algorithm 6
2:            for XR from 1 to n do
                  derive CLS1 ← ( XR1 ≪ length(XR1))
                  derive CLS2 ← ( XR2 ≪ length(XR2))
                  derive CLS3 ← ( XR3 ≪ length(XR3))
                  .....
                  derive CLSn ← ( XRn ≪ length(XRn))
```

```
        end for
3:      record CLS1, CLS2, CLS3......CLSn
        for further processing
```

The major concern in cloud computing is the entity in charge of key derivation. The TPA is granted responsibility for key derivation in this article since the TPA is a trusted authority and the whole communication between the data owner and the cloud user falls under the jurisdiction of the TPA. At the TPA site, **Algorithm 8** was developed, and it accepts a single circular left shifting block CLS and creates a set of secret keys using the SHA1 method. This approach generates a 160-bit hash tag (H1) from which 128 bits of character data are retrieved and used as a $Key_1$ which is given to cloud user $Cu_1$. For cloud users $Cu_2$, $Cu_3$,…$Cu_n$, similar methods are needed to generate the $Key_2$, $Key_{3.....}$ $Key_n$.

```
Algorithm 8   Key derivation at TPA site
Input:        CLS=CLS1, CLS2, CLS3....CLSn
Output:       Hs=H1, H2, H3...Hn
              SecKeys=Key1, Key2, Key3....Keyn
1:            receives CLS=CLS1, CLS2, CLS3....CLSn from Algorithm 7
2:            derive 160 bits' hash tag from Hs
              for Cu from 1 to n do
              derive H1 ←SHA1(CLS1)
              derive H2 ←SHA1(CLS2)
              derive H3 ←SHA1(CLS3)
              .....
              derive Hn ←SHA1(CLSn)
              end for
3:            retrieve 128 bit characters randomly from Hs
              for Hs from 1 to n do
              Key1 ←random16 (H1)
              Key2 ←random16 (H2)
              Key3 ←random16 (H3)
              .............
              Keyn ←random16 (Hn)
4:            record Key1, Key2, Key3....Keyn for further processing
```

At the TPA site, **Algorithm 9** is critical for encrypting necessary data blocks. By submitting a request to the cloud service provider, cloud users can acquire the data blocks they desire. The TPA divides the entire data file (($D_f$)) into a number of data blocks before encrypting them ($D_f$). Once the request is authorized, the TPA encrypts the requested data blocks with secrete key derived for the cloud user and sends the encrypted blocks to the relevant cloud user via secure channel, along with the exact secret key that corresponds to a cloud user. This procedure is repeated for encrypting data blocks for the author, who is a registered cloud user.

```
Algorithm 9   Blocks encryption at TPA site
Input:        Db j , SecKeys=Key1, Key2, Key3....Keyn
Output:       E(Key1(Dbj)), E(Key2(Db j))...E(Keyn(Dbj))
1:            convert Df into Db j where j=0,1,2,3....n
2:            receives request from the cloud user
              produce encrypted data blocks depends on the request
              for Db j from 1 to n do
              determine E(Key1(Dbj))
              determine E(Key2(Dbj))
              determine E(Key3(Dbj))
              ................
              determine E(Keyn(Dbj))
              end for
3:            sends corresponding encrypted blocks to the cloud user
              along with exact secret key through secure channel
```

**Algorithm 10** is designed to decrypt requested data blocks at the cloud user's location. If the cloud user receives the desired data blocks in encrypted form together with the secrete key, the data blocks can be decrypted by running D(Key(Dbj)). This step is performed for author registered cloud users in order to decrypt the desired data blocks.

```
Algorithm 10  Blocks decryption at the cloud user site
Input:        encrypted data blocks
              SecKeys=Key1, Key2, Key3....Keyn
Output:       D(Key1(Db j )), D(Key2(Db j ))... D(Keyn(Db j))
1:            if cloud user requests the data blocks from the CSP
              with the data credentials then
2:            decrypt the data blocks
              for D(Dbj) from 1 to n do
              determine D(Key1(Dbj))
              determine D(Key2(Dbj))
              determine D(Key3(Dbj))
              .............
              determine D(Keyn(Dbj))
              end for
3:            cloud user performs the operations on D(Db j)
```

### 3.1 Implementation

The proposed scheme is implemented using a Java-based JSP web application. The scheme is being tested in a physical cloud environment such as Amazon Web Services. Device setup includes Tomcat 8.5 with Corretto 11 running on 64bit Amazon Linux 2/4.1.2.

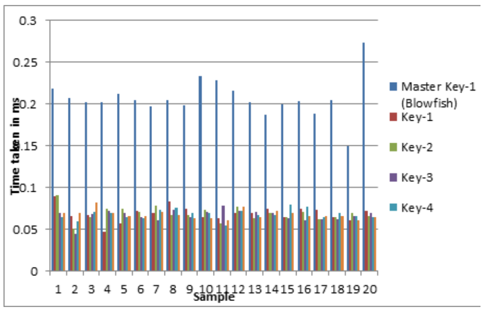
## 4. Experimental results

The suggested approach is developed and implemented to reduce computing costs while enhancing key derivation robustness. The master key$^1$ is generated using the standard Blowfish method, and this key is used to create the number of secret keys. In this study, we produced five secret keys by combining user attributes and randomly generated characters from the master key. The computational overhead that happens during key creation is seen in Table 2. The graphical representation in Figure 1 indicates that the suggested technique may incur little processing cost during the key generation process.

**Table 2 Time taken (ms) for key derivation (Blowfish vs. Proposed methodology)**

| Sample | MasKey$^1$ (Blowfish Algorithm) | Proposed method | | | | |
|---|---|---|---|---|---|---|
| | | Key$^1$ | Key$^2$ | Key$^3$ | Key$^4$ | Key$^5$ |
| 1 | 0.218 | 0.089 | 0.091 | 0.069 | 0.065 | 0.070 |
| 2 | 0.207 | 0.066 | 0.051 | 0.044 | 0.060 | 0.069 |
| 3 | 0.202 | 0.067 | 0.065 | 0.068 | 0.071 | 0.082 |
| 4 | 0.202 | 0.047 | 0.075 | 0.072 | 0.070 | 0.069 |
| 5 | 0.212 | 0.057 | 0.074 | 0.069 | 0.064 | 0.066 |
| 6 | 0.205 | 0.072 | 0.071 | 0.065 | 0.063 | 0.066 |
| 7 | 0.197 | 0.070 | 0.078 | 0.061 | 0.073 | 0.071 |
| 8 | 0.205 | 0.083 | 0.067 | 0.073 | 0.076 | 0.067 |
| 9 | 0.199 | 0.075 | 0.067 | 0.064 | 0.069 | 0.063 |
| 10 | 0.233 | 0.064 | 0.073 | 0.071 | 0.070 | 0.063 |
| 11 | 0.228 | 0.063 | 0.057 | 0.078 | 0.055 | 0.061 |
| 12 | 0.216 | 0.070 | 0.077 | 0.072 | 0.072 | 0.077 |
| 13 | 0.202 | 0.069 | 0.063 | 0.071 | 0.067 | 0.065 |
| 14 | 0.187 | 0.074 | 0.069 | 0.069 | 0.067 | 0.072 |
| 15 | 0.200 | 0.064 | 0.064 | 0.063 | 0.079 | 0.070 |
| 16 | 0.203 | 0.074 | 0.071 | 0.061 | 0.077 | 0.066 |
| 17 | 0.189 | 0.073 | 0.062 | 0.062 | 0.064 | 0.066 |
| 18 | 0.205 | 0.064 | 0.064 | 0.062 | 0.069 | 0.066 |
| 19 | 0.150 | 0.061 | 0.069 | 0.066 | 0.066 | 0.061 |
| 20 | 0.273 | 0.072 | 0.066 | 0.070 | 0.065 | 0.064 |
| **Avg.** | **0.206** | **0.068** | **0.068** | **0.066** | **0.068** | **0.067** |

**Fig. 1 Time taken (ms) for key derivation (Blowfish vs. Proposed methodology)**



The third party auditor (TPA) is in charge of converting the data file into a number of blocks, which are then requested by the intended cloud user. The TPA extracts necessary blocks from the data file, and these blocks are encrypted using the cloud user's intended keys. The computational cost of encrypting data blocks with various cloud users and their keys is shown in Table 3. The computational cost of decrypting data blocks for cloud users with related keys is depicted in Table 4.

**Table 3 Encryption time in ms with different bytes of data**

| **Data Bytes** | **200** | **330** | **470** | **650** | **820** |
|---|---|---|---|---|---|
| Key$^1$ | 0.081 | 0.082 | 0.084 | 0.160 | 0.508 |
| Key$^2$ | 0.106 | 0.111 | 0.112 | 0.212 | 2.382 |
| Key$^3$ | 0.108 | 0.109 | 0.112 | 0.213 | 0.998 |
| Key$^4$ | 0.106 | 0.109 | 0.112 | 0.202 | 1.618 |
| Key$^5$ | 0.112 | 0.119 | 0.120 | 0.228 | 5.248 |

**Table 4 Decryption time in ms with different bytes of data**

| **Data Bytes** | **200** | **330** | **470** | **650** | **820** |
|---|---|---|---|---|---|
| Key$^1$ | 0.060 | 0.069 | 0.070 | 0.078 | 0.086 |
| Key$^2$ | 0.102 | 0.110 | 0.114 | 0.375 | 1.909 |
| Key$^3$ | 0.094 | 0.096 | 0.098 | 0.107 | 0.110 |
| Key$^4$ | 0.100 | 0.101 | 0.113 | 0.415 | 3.486 |
| Key$^5$ | 0.104 | 0.106 | 0.120 | 0.137 | 3.525 |

## 5. Discussion

The proposed system's design may address the following issues in a real-time cloud environment.

### 5.1 Shannon Entropy evaluation

The Shannon Entropy is used in information security to assess the uncertainty of a random variable. In our research, this technique is used to determine the number of bits in a byte. As the number of bits' increases, so does the strength of the derived key. The Shannon Entropy of existing keys produced from the Blowfish technique and suggested method is shown in Table 5. It is discovered from the experimental data that the proposed method's uncertainty increases, which also raises the key robustness.

**Table 5: Shannon Entropy evaluation (Blowfish vs. Proposed methodology)**

| Sample | MasKey[1] (Blowfish Algorithm) | Proposed method | | | | |
|---|---|---|---|---|---|---|
| | | Key[1] | Key[2] | Key[3] | Key[4] | Key[5] |
| 1 | 2.216 | 3.030 | 2.905 | 3.125 | 3.202 | 3.155 |
| 2 | 2.216 | 3.077 | 3.280 | 2.952 | 3.077 | 3.250 |
| 3 | 2.646 | 2.858 | 2.727 | 3.030 | 3.202 | 2.680 |
| 4 | 2.646 | 2.952 | 3.202 | 2.727 | 3.327 | 3.250 |
| 5 | 2.306 | 3.327 | 2.899 | 3.202 | 2.858 | 2.780 |
| 6 | 2.872 | 2.811 | 3.155 | 3.030 | 2.952 | 3.000 |
| 7 | 2.597 | 3.031 | 3.156 | 3.024 | 2.906 | 3.500 |
| 8 | 2.700 | 3.031 | 2.899 | 3.203 | 3.078 | 2.781 |
| 9 | 2.375 | 2.727 | 2.352 | 3.203 | 2.727 | 2.953 |
| 10 | 2.647 | 3.281 | 2.875 | 3.078 | 3.031 | 3.078 |
| 11 | 2.772 | 3.156 | 2.781 | 3.328 | 2.781 | 3.203 |
| 12 | 2.597 | 2.774 | 3.078 | 2.727 | 3.250 | 2.750 |
| 13 | 2.093 | 3.078 | 2.906 | 3.031 | 2.555 | 2.686 |
| 14 | 2.772 | 3.203 | 2.774 | 3.031 | 3.024 | 3.375 |
| 15 | 2.463 | 3.203 | 2.656 | 3.250 | 2.828 | 2.727 |
| 16 | 2.597 | 3.203 | 3.156 | 2.858 | 2.983 | 2.656 |
| 17 | 2.597 | 3.031 | 3.078 | 3.108 | 2.875 | 2.656 |
| 18 | 2.463 | 2.703 | 2.983 | 2.828 | 3.328 | 2.953 |
| **Avg.** | **2.531** | **3.026** | **2.936** | **3.040** | **2.999** | **2.968** |

## 5.2 Security analysis

The key derivation process is strictly used in the suggested scheme security analysis. Each cloud user is given a unique key that only the intended cloud user and the third-party auditor (TPA) are allowed to know. The master key is obtained via the Blowfish algorithm and then transmitted to the TPA for a number of key derivations based on the available cloud user during key derivation. The TPA makes use of this master key to derive the keys by combining the appropriate cloud user characteristic and the master key. Additionally, improve critical resilience in this research by extracting random characters from the concatenated output. The SHA1 hashing algorithm is utilized to generate the keys. Random character extraction is used to increase the key's robustness.

## 5.3 Performance analysis

The proposed scheme is designed in such a way that the computational cost of deriving the keys from the master key is greatly reduced. Initially, the master key is used to encrypt/decrypt entire data blocks; later, the TPA determines the number of keys required for the cloud user.

## 5.4 Hierarchical authority security

The TPA must construct the secret key using the cloud user's attribute and the master key when the cloud user decrypts the data blocks. The output of the attribute and master key concatenation is submitted to the SHA1 hashing technique, which generates the 160-bit hash code. This 160-bit hash code is considered, and a random mathematical procedure is employed to retrieve 128 bits from the 160-bit hash code. To summarize, cloud user secret key generation is entirely dependent on individual characteristics and increases system security while minimizing overall risk.

## 5.5 Access policy flexibility

The suggested technique allows for flexibility between the TPA and the cloud user in order to produce the secret keys. The cloud user can acquire access to the data they require by interacting with the TPA. The entire process is predicated on the encryption of required data blocks with the specific keys associated with a cloud user.

## 5.6 System scalability

Initially, the system is intended to create and establish secret keys of 120 bits for a limited number of cloud users. The secret keys of 128 bits are extracted using a 160-bit hash algorithm. By increasing the hash code from 160 bits to 256 bits or more, we can simply expand the system to produce secret keys for a larger number of cloud users. Simultaneously, we may strengthen the system's overall security, which is simply reliant on the number of cloud users involved. Table 6 compares the proposed strategy to comparable research work in terms of functionality.

**Table 6 Functional comparison**

| Parameters | Jialu Hao et al. [6] | Sana Belguith et al. [8] | Mahender Kumar et al. [9] | Rashad Elhabob et al. [14] | Proposed Method |
|---|---|---|---|---|---|
| Flexible Access Control | Yes | Yes | Yes | Yes | Yes |
| Privacy policy | Yes | Yes | Yes | Yes | Yes |
| Computation cost for key derivation | High | High | High | High | Low |
| Non-Repudiation | No | No | No | No | Yes |
| Storage overhead for key derivation | High | High | High | High | Low |
| Key robustness | Medium | Medium | Medium | Medium | High |

## 6. Conclusion

We provide a low-cost access control mechanism based on attribute-based key creation in this work. The encrypted data is outsourced to the cloud by the data owner, and the third-party auditor has total control over the encrypted data (TPA). The TPA is responsible for decrypting the data provided by the data owner and partitioning the entire data file into blocks. The secret key, which is derived from the attribute weight and random characters obtained from the master key, is used to encrypt each block. The cloud user can access the data in the cloud after successfully enrolling. The system is designed to provide hierarchical key distribution to each cloud user. The numerical findings reveal that the proposed system has lower processing costs and more consistent key distributions. The suggested solution may provide flexible data access control while also preventing unauthorized data access.

## References

1. P. Kumar, P. Alphonse et al., “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.

2. K. Albulayhi, A. Abuhussein, F. Alsubaei, and F. T. Sheldon, "Fine-grained access control in the era of cloud computing: An analytical review," in 2020 10th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2020, pp. 0748–0755.
3. N. Garg, S. Bawa, and N. Kumar, "An efficient data integrity auditing protocol for cloud computing," Future Generation Computer Systems, vol. 109, pp. 306–316, 2020.
4. S. Ghosh and V. Karar, "Blowfish hybridized weighted attribute-based encryption for secure and efficient data collaboration in cloud computing," Applied Sciences, vol. 8, no. 7, p. 1119, 2018.
5. Q. Tian, D. Han, and Y. Jiang, "Hierarchical authority based weighted attribute encryption scheme," Computer Science and Information Systems, vol. 16, no. 3, pp. 797–813, 2019.
6. J. Hao, C. Huang, J. Ni, H. Rong, M. Xian, and X. S. Shen, "Fine-grained data access control with attribute hiding policy for cloud-based iot," Computer Networks, vol. 153, pp. 1–10, 2019.
7. S. Xu, J. Ning, Y. Li, Y. Zhang, G. Xu, X. Huang, and R. Deng, "Match in my way: Fine-grained bilateral access control for secure cloud-fog computing," IEEE Transactions on Dependable and Secure Computing, 2020.
8. S. Belguith, N. Kaaniche, A. Jemai, M. Laurent, and R. Attia, "Pabac: a privacy preserving attribute based framework for fine grained access control in clouds," in SECRYPT 2016: 13th International Conference on Security and Cryptography, vol. 4. SciTePress, 2016, pp. 133–146.
9. M. Kumar and S. Chand, "Eski-ibe: Efficient and secure key issuing identity-based encryption with cloud privacy centers," Multimedia Tools and Applications, vol. 78, no. 14, pp. 19 753–19 786, 2019.
10. S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," IEEE Transactions on Information Forensics and Security, vol. 13, no. 8, pp. 2101–2113, 2018.
11. Y. Liu, Y. Zhang, J. Ling, and Z. Liu, "Secure and fine-grained access control on e-healthcare records in mobile cloud computing," Future Generation Computer Systems, vol. 78, pp. 1020–1026, 2018.
12. F. Cai, N. Zhu, J. He, P. Mu, W. Li, and Y. Yu, "Survey of access control models and technologies for cloud computing," Cluster Computing, vol. 22, no. 3, pp. 6111–6122, 2019.
13. K. Shivanna, S. P. Deva, and M. Santosh Kumar, "Privacy preservation in cloud computing with double encryption method," in Computer Communication, Networking and Internet Security. Springer, 2017, pp. 125–133.
14. R. Elhabob, Y. Zhao, N. Eltayieb, A. M. Abdelgader, and H. Xiong, "Identity-based encryption with authorized equivalence test for cloud-assisted iot," Cluster Computing, vol. 23, no. 2, pp. 1085–1101, 2020.
15. K. Lee, "A generic construction for revocable identity based encryption with subset difference methods," PloS one, vol. 15, no. 9, p. e0239053, 2020.
16. H. Cheng, C. Rong, M. Qian, and W. Wang, "Accountable privacy-preserving mechanism for cloud computing based on identity-based encryption," IEEE Access, vol. 6, pp. 37 869–37 882, 2018.
17. C. Meshram, C.-C. Lee, S. G. Meshram, and M. K. Khan, "An identity-based encryption technique using subtree for fuzzy user data sharing under cloud computing environment," Soft Computing, vol. 23, no. 24, pp. 13 127–13 138, 2019.
18. H. Deng, Z. Qin, Q. Wu, Z. Guan, R. H. Deng, Y. Wang, and Y. Zhou, "Identity-based encryption transformation for flexible sharing of encrypted data in public cloud," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3168–3180, 2020.
19. H. Li, Y. Dai, and B. Yang, "Identity-based cryptography for cloud security," Cryptology ePrint Archive, 2011.
20. J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," Ieee Transactions on computers, vol. 64, no. 2, pp. 425–437, 2013.
21. N. Farjana, S. Roy, M. Mahi, J. Nayeen, and M. Whaiduzzaman, "An identity-based encryption scheme for data security in fog computing," in Proceedings of International Joint Conference on computational intelligence. Springer, 2020, pp. 215–226.
22. Q. Huang, W. Yue, Y. He, and Y. Yang, "Secure identity-based data sharing and profile matching for mobile healthcare social networks in cloud computing," IEEE Access, vol. 6, pp. 36 584–36 594, 2018.
23. X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," Information Sciences, vol. 494, pp. 193–207, 2019.
24. P. Mishra and V. Verma, "Study of identity-based encryption for cloud data security," in Decision analytics applications in industry. Springer, 2020, pp. 401–408.
25. P. Sharma, N. R. Moparthi, S. Namasudra, V. Shanmu-ganathan, and C.-H. Hsu, "Blockchain-based iot architecture

to secure healthcare system using identity-based encryption," Expert Systems, p. e12915, 2021.

26. D. Unal, A. Al-Ali, F. O. Catak, and M. Hammoudeh, "A secure and efficient internet of things cloud encryption scheme with forensics investigation compatibility based on identity-based encryption," Future Generation Computer Systems, vol. 125, pp. 433–445, 2021.
27. N. Vaanchig, Z. Qin, and B. Ragchaasuren, "Construct-ing secure-channel free identity-based encryption with equality test for vehicle-data sharing in cloud comput-ing," Transactions on Emerging Telecommunications Technologies, p. e3896, 2020.