Achieving A Secure Cloud Storage Mechanism Using Blockchain Technology

Parin Patel* and Hiren Patel

***Abstract*—Storage requirements are increasing rapidly with fast-growing Cloud computing technology. The more data storage and transaction processing requirements, the more chances of malicious activities. Storage security is a vital concern nowadays. To deal with this problem, in this research, we propose a system that provides an efficient security mechanism for data storage on the Cloud with enhanced access control policies by using Blockchain technology, which is immutable in nature. The usage of smart contracts improves transparency and creates a trust model that eliminates the requirement of a trusted third party. The system manages the log trails of all transactions and accomplishes all access policies. The proposed model is transparent, traceable, and secure. Data owners hold all rights to their data and any activity pertaining to accessing the data illegitimately shall not be accepted by the network and shall be notified to all the concerns. We demonstrate our proposed mechanism with details about all concerned stakeholders. The Attribute-Based Encryption (ABE) scheme is used with multiple authorities to manage the user attributes that avoid the use of the central authority. The smart contract avoids the computational burden on the user side and reduces communication costs. We analyze security and compare the performance of our proposed model.**

***Index Terms*—Cloud storage, data sharing, blockchain, smart contract, access control, attribute-based encryption, integrity**

I. Introduction

Advancements in communication and computing technologies, inexpensive data rates, and increased usage of portable devices have led to an escalation in online data storage over the Internet. According to the latest study, Walmart handles transactions of more than 1 million clients every hour, estimated to be more than 2.5 petabytes of information—the equivalent of 167 times the books in America's Congress Library [1]. Stored data plays an essential role in future plans, advertisements, service illustrations, etc. Cloud computing is the technology that is used to remotely store your data on a pay-as-you-use basis without worrying about managing the hardware or other resources. However, due to the lack of direct owner's control over the data, privacy, and security have become the primary concerns in adopting Cloud computing.

Cloud computing provides services on user's demand mostly on a rental basis, without worrying about managing the resources. It provides many potential services like on-demand services, huge data storage, etc. Due to this scenario, many technologies are invented to handle this mass storage access over the network. Along with huge data storage, concerns such as security, availability, reliability, failure recovery, and on-demand access are also important nowadays. Cloud computing offers huge virtual data storage with solutions to these concerns.

As the data owner stores data on Cloud storage which is a non-trusted third party, issues arise such as privacy and security due to loss of direct control over the data. Along with this issue of data ownership, there are issues of data privacy and access control policies where part of the data is permitted to access (a group of) selected user(s). Apart from these Cloud-specific issues, other traditional networking issues such as various types of attacks, malware, etc. remain the concerns. Furthermore, Cloud consumers expect the Cloud service providers to address the solutions to the issues such as data sharing, data leakage, authentication to legitimate uses, backup, unauthorized access, etc., Hence, it becomes the sole responsibility of Cloud service providers to build a trust model with its user by offering them satisfactory responses to their apprehensions.

The idea of Blockchain was circuitously invented by an individual (or group of people) using the name Satoshi Nakamoto in 2008 to serve as the general public transaction ledger of the cryptocurrency viz. bitcoin [2]. Blockchain technology permits distributed public ledgers that hold immutable information through a secure and encrypted method and makes sure that transactions/data stored on it will never be altered. It is the technology that allows all users to keep a replica of the common ledger containing all transactional data and update it to maintain consistency among them. Unlike Cloud, there is no central authority in Blockchain as it offers distributed and decentralized environment. In absence of a trusted third party, the transactions in a block are validated through a group of selected nodes (sometimes also known as miners) which makes it error-proof.

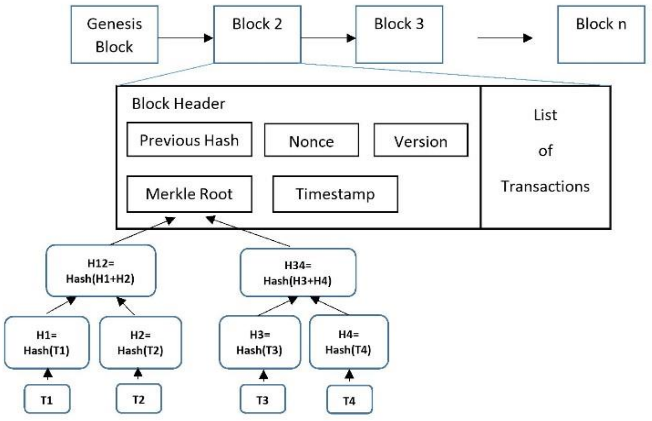

Fig. 1. Block structure in blockchain technology.

Manuscript received November 25, 2022; revised February 14, 2023; accepted May 6, 2023.

Parin Patel is with Kadi Sarva Vishwavidyalaya, Gandhinagar 382015 Gujarat, India.

Hiren Patel is with Sarva Vidyalaya Kelavani Mandal, Kadi 382715, Gujarat, India.

*Correspondence: parinpatel1787@gmail.com (P.P.)

Blockchain offers various features such as immutability, security, transparency, and traceability through existing mechanisms such as public distributed ledger, hashing, encryption, digital signature, and transaction validation through a special mechanism called Proof of Work.

As shown in Fig. 1, Blockchain is a chain of blocks where each block contains the block header and set of transactions. Block header contains previous block hash value and Merkle root, time stamp, version, nonce, etc. Merkle tree is built continuously by hashing a pair of nodes until there is only one hash left. In the Merkel tree, every leaf node is a transaction and every non-leaf node is a hash of its child nodes.

Each block has a timestamp which makes it harder for an adversary to manipulate. They also make it harder for an adversary to manipulate the Blockchain as well as serve as a source of variation for the block hash. Nonce (number used only once) is a number added to a hashed block that meets the limitations of difficulty level when rehashed. The nonce is to be guessed through computations by Blockchain miners. A transaction must be added to the block after it is validated by the miner(s). For example, a valid transaction implies Bob got one bitcoin from Alice. Alice may have attempted to pass the same bitcoin to Carol, however, since it is a digital asset. Nodes must therefore reach an agreement on which transactions must be maintained in the Blockchain to ensure that no corrupt branches and divergences exist [3, 4]. In fact, this is the objective of the second layer of Consensus. There are different consensus mechanisms depending on the type of Blockchain [5]. Proof-of-work (PoW) is the most well-known. To ensure authentication and verifiability, PoW requires solving a complicated computational process such as finding hashes with specific patterns, e.g., a leading number of zeroes [6]. Instead of dividing blocks proportionally to the relative hash rates of miners (i.e., their mining power), protocols such as Proof-of-Stake (PoS) divide trust blocks proportionally [7]. Thus, the choice is fairer and keeps the network from being dominated by the richest participant. Because of the significant reduction in power consumption and enhanced scalability, Blockchain, such as Ethereum [8], gradually shift to PoS. Byzantine Fault Tolerance (BFT) [9] and its variants [10] are other consensus methods.

In the traditional way of transferring money, people meet each other and transfer it. Rather than meeting in person, it has started providing digital money to each other by using the services of the internet. Alice sends digital cash money to Bob. Bob is not receiving digital money. Bob can’t be sure whether Alice had sent digital money to him or it could be a case that Alice tries to send her digital money to two different people at the same time. Alice could send digital money to Bob and Dev. To make sure that this type of scenario does not happen in transactions. They decide one single person can note down all the transactions in the book. This makes sure their network is secure and people who transfer money can’t cheat. This is the process of how our banks work. Soon the network becomes huge, it may be possible that Dev becomes a money holder and tries to change some transactions or manifest fake transactions for his benefit. To avoid such problems, all the members of the group hold a copy of all transactions. If someone tries to alter a transaction, then everyone in the network can verify it. Noting down all the transactions in the ledger makes sure that double spending is not allowed. Moreover, there is no need to rely on trust as there is no centralized system. Trust is developed through consensus where everyone is connected to the network and makes sure that the system works fruitfully.

The server (containing hosted applications) is prone to attack such as a single point of failure. Backup server results in cost and maintenance overhead. In the Blockchain, there is no single server. There are multiple servers in the form of various nodes which are hosted with a completely synced Blockchain or database. Any data in the Blockchain is shared with everyone who participates and it is made available on all nodes. This helps to remove central authority and overheads related to backup servers and databases. We depend on the authorities to resolve our issues. Blockchain provides trust using a cryptographic measure.

In this paper, we have studied the solution provided to security issues of Cloud storage using Blockchain technology. The rest of the paper is arranged as follows: in Section II, we discuss related work, which addresses contemporary work in terms of security issues in Cloud storage using Blockchain technology. Section III describes some preliminary studies related to our proposed work. We present our system with an analysis in Section IV. Further, a detailed study regarding security, functional, and performance analysis is mentioned in section VI followed by the conclusion and future work in Section VII.

Table I contains the system notations that are used in the research paper.

TABLE I: SYSTEM NOTATIONS

Parameter	Description
DO	Data Owner
DU	Data User
CSP	Cloud Service Provider
TRL	Transaction Log
ACL	Access Control Log
AES	Advanced Encryption Standard
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CA	Certification Authority
AA	Attribute Authority
Customer ID	Unique User Identification for each user on Cloud
H_F	Hash of Data File F
K_s	Common Shared Secret Key ‘S’
$P_{R\text{-}DO}$	Private Key of Data Owner
$P_{U\text{-}DO}$	Public Key of Data Owner
$P_{R\text{-}DU}$	Private Key of Data User
$P_{U\text{-}DU}$	Public Key of Data User
FileID	Unique File Identification for each data file on Cloud
BC	Blockchain

II. Related Work

In this section, we put forth the contemporary work being carried out in the domain of Cloud data storage security through the use of Blockchain Technology. And based on this study, we identified research challenges in the current work and proposed our work in the next section to address the challenges.

Li and Wu *et al.* [11] have proposed Block-secure, which is Blockchain-based secure P2P Cloud storage that provides security over the data. In their solution, researchers have used Blockchain to store data like file URLs, File replica URLs, Hash, Transaction details, and keys. The Authors used a customized genetic algorithm to improve the performance of distributed architecture. They have compared multiple users with a single data center and multiple users with multiple data centers using Blockchain and a genetic algorithm. The file loss ratio is claimed to be low. Sukhodolskiy and Zapechnikov [12] have proposed a Blockchain-based access control system using a smart contract which provides more security on data. In this system, an enhanced attribute-based encryption scheme is used to manage the access control mechanisms with a smart contract. The file is encrypted by the owner and all details like public link, hash code, and access policies are stored in the contract files. Ethereum virtual machine manages all contracts and certifications to access, edit and delete permission for the file. Ramamoorthy and Baranidharan [13] have proposed a framework that provides more security and over-data access and modification. The use of Blockchain provides more security for data and also restricts any malicious activities. Xue and Xu *et al.* [14] have proposed Dstore that manages lease relationships using a smart contract. It verifies the correctness of data and stores it for the payment process and also manages lease relationships automatically without the interference of any third party. Liang and Shetty *et al.* [15] have proposed a system that uses a Blockchain-based auditing system for data provenance called Provchain. Every transaction is recorded with an immutable timestamp and generates a receipt for each data for validation. The system provides transparency, privacy, and reliability to the users. Yue and Li *et al.* [16] have proposed a framework that uses the Merkle tree and sampling verification for integrity verification. The use of a sampling strategy increases the performance of verification. It removes distrust that exists in a traditional environment using Blockchain and sampling. Wei and Wang *et al.* [17] have proposed a "block-and-response system to verify the integrity of data. Smart contract monitors any data change using the Merkle hash tree. Automatically user gets a warning message when the data is accessed from the Cloud. This system manages integrity verification easily and gives a fast response. Omar and Bhuiyan *et al.* [18] have proposed a MediBchain that manages data stored in Cloud using Blockchain technology. A privately accessible unit manages communication between users and the Blockchain. All metadata is stored on the Blockchain and encrypted patient data is stored in Cloud. Li and Wu *et al.* [19] have proposed a novel scheme that removes deduplication using Blockchain technology and also processes missing or altered file recovery. The use of smart contracts does automatic transactions without the use of a third party. Kumar and Singh *et al.* [20], Pourmajidi and Miranskyy [21], Sutton and Samavi [22] have proposed solutions that provide immutable log storage using Blockchain technology. In their systems, all transactions are stored on a block of Blockchain that is immutable and tamper-proof. Qin and Huang *et al.* [23] proposed a BMAS Blockchain-based multi-authority Access Control scheme that reduces single-point failure of the CP-ABE scheme and uses the advantage of Blockchain technology. This scheme uses Shamir's secret sharing scheme and permissioned Blockchain. Wang *et al.* [24] have provided access to data used for a particular period and it provides tracing of the access control system. It uses smart contracts to communicate with the system and the data owner stores cipher data on the Blockchain. Yang and Chen *et al.* [25] have used CP-ABE to hide access policies. In this paper, they have used only attribute authority to manage public key shares to users. There is data accessing user generates the key for decryption and storage access communication. In this paper, the data owner performs all the operations and uploads values to the Blockchain network. Li [26] have proposed a system with verifiable access policies using Blockchain technology. They implemented a data exchange network by assigning unique IDs to nodes and using an access mechanism. The data user verifies using a signature stored on the Blockchain network and provides access details of the URL and public key. Zhao and Zhang *et al.* [27] used a Java-based Blockchain network to register a client with attribute authority. Attribute modification will be handled by a fine-grained access control mechanism. CP-ABE scheme is used to manage the key generation mechanism. Sammy and Vigila [28] have implemented a patient health record CP-ABE scheme using Elliptic Curve Cryptography (ECC) with a hierarchical access structure. Dynamic attributes are used to manage access control to allow multiple authorities for data access. Xue *et al.* [29] implemented a dynamic access control scheme on cloud storage to provide the most secure communication. Ciphertext-policy attribute-based encryption is the most secure policy in data access control mechanisms on public cloud storage. Hao and Huang *et al.* [30] implemented a fine-grained data access control policy with an attribute hiding policy on cloud storage with IoT. It shows that an access control system gives faster communication with a secure channel. Lewko *et al.* [31] introduced Multi authority Attribute-based Encryption scheme with Boolean formula. They developed the scheme which prevents the system from collusion attacks. Benil *et al.* [32] utilized Certificate less Aggregate Signature (CAS) technique to create the digital signature and Elliptic Curve Cryptography (ECC) to encrypt medical data before sharing and storing it in cloud storage. Wang *et al.* [33] implemented blockchain-based fair payment smart contract for cloud storage. Smart contracts are used for the payment system. Sharma *et al.* [34] implemented a system with the Ciphertext Policy Attribute-Based Encryption (CP-ABE) algorithm it used user revocation methods that provide access control and in the cloud storage system.

To deal with the challenges and problems mentioned in the above research papers, we proposed a scheme wherein we compute the attributes and tokens through a smart contract. Further, the cryptographic operations are performed through the smart contract, which results in the creation of trust among the non-trusted entities and offers a reduction in computational and communication overhead. We proposed a novel solution in the access control scheme using an updated ABE scheme using Blockchain technology. We used a smart contract to overcome the issue of central authority or third party. Our contributions through this paper are as follows.

1) To maintain confidentiality and to keep control of data with the owner, we store the encrypted data on the Cloud server hence protecting data confidentiality.
2) Usage of public key cryptography avoids the usage of transferring private keys over insecure public channels. Further, a Certification Authority (CA) helps to avoid ownership concerns.

3) Legitimate users with a valid certificate from the Certification Authority (CA) can only participate in the entire process.
4) Updated Attributed-Based Encryption (ABE) is used to provide an access control mechanism to restrict immeasurable access over any data by any user.
5) Option of rights revocation enables us to survive in a dynamically changing environment.
6) Smart contracts are used which are automatic and even-triggered in nature that results in faster communication.

III. Preliminaries

In this section, we analyze schemes that we have proposed in Figs. 2 and 3 such as Bilinear maps, Multi-Authority Ciphertext Policy-Attribute Based Encryption (CP-ABE), and Blockchain.

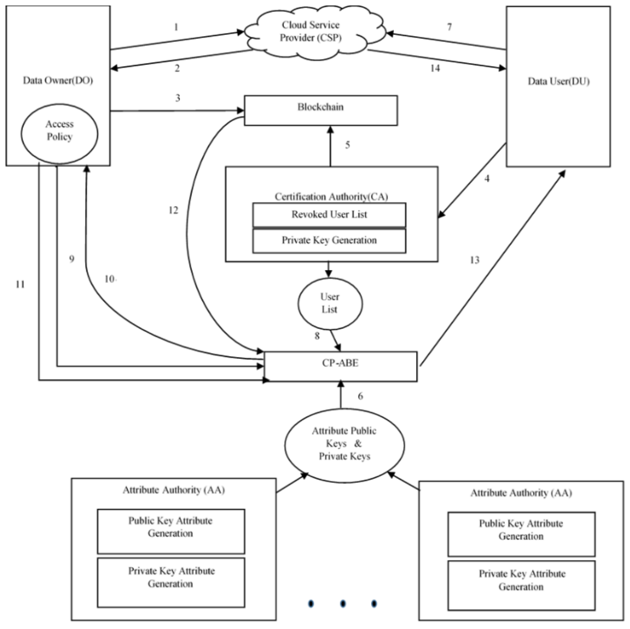
Data Owner(DO)
Access Policy
Cloud Service Provider (CSP)
Data User(DU)
Blockchain
Certification Authority(CA)
Revoked User List
Private Key Generation
User List
CP-ABE
Attribute Public Keys & Private Keys
Attribute Authority (AA)
Public Key Attribute Generation
Private Key Attribute Generation
Attribute Authority (AA)
Public Key Attribute Generation
Private Key Attribute Generation
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Fig. 2. Access control system model using blockchain.

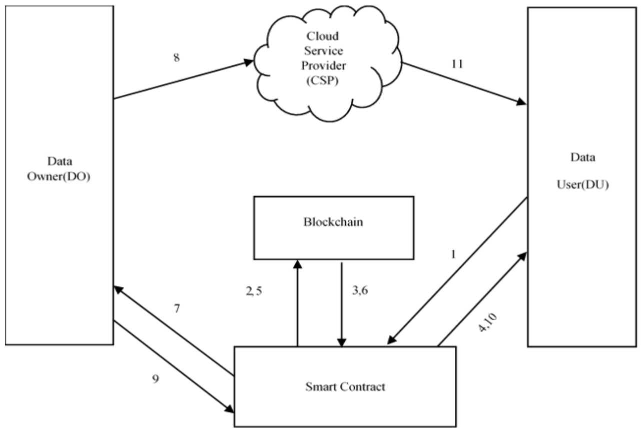

Fig. 3. File integrity verification and recovery system model.

A. Bilinear Maps

A Bilinear map is used to map any attributes to a random element of GT. We use this mapping in our multi-authority attribute-based encryption scheme. A Bilinear map e: G × GT → GT, where G is a Gap Diffie-Hellman (GDH) group and GT is multiplicative cyclic groups of prime order p, and g is the generator of G. with the following three properties [24].

1) Computable: $g, f \in$ G and $x, y \in Z$, $e(g^x, f^y) = e(g, f)^{xy}$.
2) Bilinear: for all u1, u2 ∈ G and a, $b \in Zp$, e (ua1, ub2) = e (u1, u2) ab.
3) Non-degenerate: $e\ (g, f) \neq 1$, where g and f are generators of G.

B. Multi Authority CP-ABE Scheme

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme is decentralized and controls access policies of encrypted data in the Cloud environment. This scheme is basically composed of Setup, Encrypt, Key Generation, and decrypting. We referred definition from Lewko and Waters [31]. We use this scheme for the access control mechanism in our system. We enhance this scheme as per our requirements. Here GP is a global parameter, Private Key SK, Public Key PK, Message M, Ciphertext CT, access matrix (A, ρ), identity parameter GID, and key K.

GlobalSetup (λ) → GP. It takes security setting λ as input and provides global system parameters as output.

Authority Setup (GP) → (SK, PK). Each and every authority runs this algorithm and takes input as a global parameter and generates a Private Key and Public Key pair (SK, PK).

Encrypt (M, (A, ρ), GP, PK) → CT. This Algorithm takes message m, (A, ρ) as the access matrix, PK as the Public Key, and GP Global parameters. It generates ciphertext CT as output.

KeyGen (GID, GP, ω, SK) → (Kω, GID). This algorithm takes GID as identity, GP as a global parameter, ω as an attribute of any authority, and SK as the Private Key of the authority. It generates (Kω, GID) a pair of attributes and identities.

Decrypt (CT, GP, (Kω, GID)) → M. This algorithm takes CT as ciphertext generated from Encrypt (M, (A, ρ), GP, PK) algorithm, GP as a global parameter, a pair of attributes, and identity (Kω, GID). It generates message M if it satisfies the access matrix and attributes to ciphertext otherwise decryption fails.

C. Blockchain

Blockchain technology permits distributed public ledgers that hold changeless knowledge through a secure and encrypted method and makes sure that transactions will never be altered. It contains data or a set of transactions in a block that can never be altered. It is the technology that allows all users to keep a replica of a ledger containing all transaction data and update it to maintain the integrity of each and every transaction. Another advantage of Blockchain is it doesn't allow central authority which is there in Cloud. In the same way, Blockchain doesn't rely on a third party when each and every member of the Blockchain needs to validate the data. It is somewhat difficult to hack. Blockchain provides tracking of all transactions from first to last.

The Ethereum platform is used to design decentralized services based on Blockchain technology. It allows smart contracts and distributed applications to run and be built without any control of a third party. It is not only a platform but also uses Turing's complete programming language to build and run decentralized applications. It is an open-source platform that uses Blockchain technology and its own cryptocurrency.

We use Blockchain technology because the transaction is signed by the owner and submitted to the Blockchain, it is added to the Blockchain only if it approves by a consensus algorithm, and Transactions added to the Blockchain can never be altered or deleted by anyone. So, our system provides transparency and reliability.

We use smart contracts because it manages all data access mechanisms, and use the ownership certificate. It also manages all transaction details on the Blockchain network. All the transactions from first to last are saved on a Blockchain network that can’t be modified. A smart contract that uses less computational power. They are easy to build and deploy. Smart contract performs all operations between Cloud and Data Owner or User without the interference of third-party. A smart contract helps the user to store metadata on the Blockchain when any type of update occurs.

D. Shamir Secret Sharing Scheme (SSS)

This scheme is used to share data secretly among the network with *e* number of participants. It consists of the following two algorithms.

1) Share: $s \in Zp$ with threshold t among e participants. Build a (t-1) degree polynomial $g(i)= x_0 + x_1 i+\ldots+ x_{n-1} i^{t-1}$, where $x_0= s$, $x_1, \ldots, x_{t-1}$ are chosen randomly from Zp.Then shares with e participants as $s_n = g(i_a)$ (a = 1, 2, …, e) where $i_a \in Zp$.
2) Reconstruct: Lagrange interpolation is used to recover secret s from any t of the n shares by using the following equation.

$$s=\sum_{i=1}^{t-1} s_i \prod_{n=0,\ n\neq m}^{t-1} \frac{x_n}{x_n-x_j}$$

IV. System Model

This section provides a detailed overview of our proposed system model. We describe all participants, the flow of our proposed system model, and the access control phases of the system model. All the phases with the detailed algorithms are described in this section.

A. Entities of System Model

Critical components of the architecture are discussed as follows.

Data Owner (DO): The data owner is an entity that stores the data on the Cloud. The data owner first encrypts the data and stores it on the Cloud. Only encrypted data is stored on the Cloud whereas other information such as credentials and metadata is managed by the data owner. The metadata like keys, file URLs, and hash values are stored on the Blockchain using a smart contract.

Cloud Service Provider (CSP): Cloud Service Provider (CSP) provides Cloud storage services and maintenance of data. CSP does not have any information on metadata like keys, hash values, or URLs.

Data User (DU): Data users can request data access to CSP and according to the rights provided by the Data Owner, Data User can access the data.

Access Control Mechanism (CP-ABE): This mechanism helps to manage access control over the data. The policy defines the rights to create, update and revoke. Attributes-based access control also known as policy-based access control defines access control rights granted to users through the policy combined with attributes together. The policy can use any type of attribute like user attributes, resources, objects, environment attributes, etc. First, the Cloud User sends a request to access the data then the Cloud service provider sends a request to a smart contract. Smart contract manages the access mechanism using the ABE scheme and it allows to access the data to the requested user. The enhanced Attributes-based access control ABE scheme manages all certification and access control mechanisms using a smart contract.

Certification Authority (CA): Certification Authority creates the contract for every new user and generates Private and Public Keys for interacting with the system. Any further interaction with the system is only permitted through this contract file. CA contract refers to the user contract. Keys are transmitted with the user contract in encrypted form.

Attribute Authority (AA): Attribute Authority issues all attributes to data users through Blockchain. It generates keys for all attributes and sends a registration request to Certification Authority (CA). The certification Authority confirms the registration and certificate data in the AA contract.

Blockchain: Attribute Authority, Certification Authority, Data user, and Data Owner are entities participants in the Blockchain network which provides immutable and verified data to every user.

B. Architecture Overview

An overview of the architecture is illustrated in Fig. 2 and Fig. 3.

An overview of the Access Control System Model is illustrated in Fig. 2. All the steps of the model are described below.

1) Data Owner (DO) generates the key K_s to be used in the Advanced Encryption Standard (AES) algorithm and encrypts the file F and generates encrypted file F'. DO keeps the key Ks privately and does not share them with anyone except any legitimate DU that requests DO for data access permission.
   DOEncryption (K_s, F) $\geq$ F'. Store the F' file on a cloud server.
2) Cloud service provider (CSP) replays with the location of the file and hash key H_F of the file. Replay (URL, H_F).
3) UploadDO contract generates a token with file location URL, access policy (A, ρ), hash key H_F, and stores on Blockchain BC. (URL, (A, ρ), H_F, FileId, OwnerId) $\geq$ BC.
4) Data User (DU) sends a request to the certification authority to join the system.
5) Certification authority takes GP as a global parameter and generates public and Private Key pair ($P_{R\text{-}DU}$\*, $P_{U\text{-}DU}$\*) for Data User to join the system and upload the Public Key on the Blockchain with a user id (UID) and sends a Private Key to Data User. (GP) $\geq$ ($P_{R\text{-}DU}$\*, $P_{U\text{-}DU}$\*). Add user UID to the User list.
6) Attribute authority takes global parameter GP, User id UID, attribute belongs to that user, and secret key $P_{R\text{-}DU}$\* as input and generates the secret key $P_{R\text{-}DU\ (a,\ UID)}$ and Public Key $P_{U\text{-}DU\ (a,\ UID)}$ for system interaction. (GP, UID, $P_{R\text{-}DU}$\*, a) $\geq$ ($P_{R\text{-}DU\ (a,\ UID)}$, $P_{U\text{-}DU\ (a,\ UID)}$). Store a Public Key

on the Blockchain and send the Private Key to the data user.

7) Data User (DU) sends a request to access the file on Cloud.
8) Access control contract verifies the user from the user list. If the user id is correct then it proceeds to generate tokens to access the file from the cloud server.
9) Access control contract verifies the access policy of the file for the user provided by the Data Owner (DO) and provides access to the file for a particular time period.
10) Ask the Data Owner to provide a key K_s for a particular File Id.
11) Receive key K_s from the Data Owner.
12) Access control contract generates a token to access the file for the user. It takes the location URL, access policy (A, ρ), hash key H_F, and User id UID from Blockchain which was stored previously by the data owner. Key K_s provided by the Data Owner will encrypt first with a Public Key $P_{U\text{-}DU}$* of the user-generated from certification authority PUK*. Encrypt (K_s, $P_{U\text{-}DU}$*) $\geq$ PUK*. Token_generation (URL, (A, ρ), H_F, PUK*, UID) $\geq T_{UID}$. Encrypt the token with Public Key $P_{U\text{-}DU\ (a,\ UID)}$ generated from Attribute authority. Encrypt (T_{UID}, $P_{U\text{-}DU\ (a,\ UID)}$) $\geq ET_{UID}$.
13) Data User (DU) receives token using smart contract Decrypt (ET_{UID}, $P_{U\text{-}DU\ (a,\ UID)}$) $\geq T_{UID}$. The data user decrypts the token using a Private Key generated from the certificate authority and receives the URL, the Encrypted key PUK* of the file to decrypt it. Decrypt (ET_{UID}, $P_{U\text{-}DU\ (a,\ UID)}$) $\geq T_{UID}$. Decrypt a key PUK* received after decryption of the token using a Private Key generated from the Attribute Authority. Decrypt (PUK*, $P_{U\text{-}DU}$*) $\geq$ PUK.
14) Data User (DU) accesses the file from the received URL and decrypts the file using the key received K_s.

C. File Integrity Verification

An overview of the File integrity verification and recovery System Model is illustrated in Fig. 3. Every step of the model is described below with an algorithm. As shown in Fig. 2, the accessing and requesting of files process is completed, and then the file verification steps 1 to 11 are completed as per Fig. 3.

1) Data User (DU) sends requests for file integrity verification. Smart contract generates a new Hash of the file NHF.
2) Smart contract gets HF of the File using FileId from Blockchain.
3) Receives the HF for a particular Field and compares both hashes.
4) If both hashes are equal, then send a verification certificate to the Data User (DU).
5) If the result is false, then it searches for OwnerId and URL using FileId from Blockchain.
6) Receives OwnerId and URL using FileId from Blockchain.
7) Sends a request to update the file with FileId and URL to Data Owner (DO).
8) Data Owner (DO) updates the file.
9) Data Owner (DO) sends a confirmation to a smart contract.
10) Smart contract sends a file update message to Data User (DU).
11) Data User (DU) again downloads the file and does the procedure of integrity verification again.

D. System Construction

Our secure Cloud storage model with an enhanced CP-ABE scheme using Blockchain is used to achieve decentralization of storage and integrity verification. Interaction between Data User and Data Owner maintained by smart contract policy using Ethereum. In the proposed scheme file access operation is managed by a smart contract and stored the file logs on the Blockchain network. The system uses the access control policy to manage all operations on the file.

E. Concrete Construction

Global Setup. There are m attributes in the system denoted as U= {1,2, 3, …, m}. A Bilinear map e: $G\times G_T \rightarrow G_T$, where G and G_T are two multiplicative cyclic groups of prime order p, and g is the generator of G. Cryptographic hash function H:{0,1}* $\rightarrow G_0$ which maps attribute to a random element of G_0.

Our system is divided into five phases.

1) Phase 1: Attribute authority setup

Attribute Authority (AA) is managed by a Shamir Secret Sharing Scheme (SSS) to obtain sub-keys. Suppose an attribute μ is managed by $n\mu$ AAs, where every AA_K selects two random numbers αk and βk as its inputs for two random polynomials $f_k(x)$ and $h_k(x)$ of $(t_\mu-1)$ degree. The Secret key is defined as

$$P_{R-DU_{k,\mu}} = P_{R-DU_{\alpha\ \mu,K}} = \sum_{i=1}^{n_\mu} S_{k,i}\,, S_{\beta\ \mu,k} = \sum_{i=1}^{n_\mu} S'_{k,i}\,\alpha_{\mu}$$

$$= \sum_{k=1}^{n_\mu} \alpha_k \qquad\qquad \beta_\mu = \sum_{k=1}^{n_\mu} \beta_k$$

AAK calculates secret keys and Public Keys.

$$P_{R-DU_{k,\mu}} = e(g,g)^{P_{R-DU_{\alpha\mu,}}{}^{i}}, g^{P_{R-DU_{\beta\ \mu,i}}}$$

This public key can be shared with any other entity.

Every time when Data user wants to communicate with the system, Attribute Authority (AA) will select a random attribute provided by the data user and generates a public key and Secret key as per the above process.

2) Phase 2: Data user setup

A Data User sends a request to the Certification Authority (CA) for registration to join a system. A Data User gets the user id UID and adds the user id UID to the user list for further communication to the system.

3) Phase 3: Encryption

The Data Owner uploads the encrypted file on a Cloud server and receives the URL of the file with the file hash value. Data Owner defines the access structure to access the encrypted file F′ with attribute-based encryption and generates the matrix (A, ρ). A denotes a × l matrix where the function ρ maps the x_{th} row of A to an attribute $\rho(x)$ ϵ At_1, …, At_i. File location URL, access policy (A, ρ), and hash key H_F

stores on Blockchain BC. For each row of A, a data owner chooses any random vector $v= (s, x_1, x_2, \dots, x_n)$ where s is a secret number and $x_1, x_2, \dots, x_n$ is used to share secret number s. Let $\lambda_{x=} A_x\ v^t$, where A denotes the x_{th} row of matrix A. It computes cipher text $E_{x=} A_x\, u^t$, where it selects any random vector u and any random number r from sets.

$$E_{PK\ a,UID} = (E_0 = PK_{a,UID}\ .\ \ e\ (g,g)^s\ ,\ E_{1,x=}\ \ e(g,g)^{\lambda_x}\ .\ e(g,g)^{\alpha_{\rho(x)}r_x}\ , E_{2,x=}\ g^{r_x}\ , E_{3,x=}g^{c_x}.(g,g)^{\beta_{\rho(x)}r_x})$$

Finally, $E_{PK\ a,UID}$ is the encrypted token and sends it to the data user to access the file from the Cloud server.

4) Phase 4: Data user access setup

When a data user wants to access the file then it sends a request to the certification authority CAKey contract generates Public Key $P_{U\text{-}DU}$* and Private Key $P_{R\text{-}DU}$*. Add User id UID to the user list. AAkey contract verifies the user id from the user list, generates the Private Key $P_{R\text{-}DU(a,\ UID)}$ and Public Key $P_{U\text{-}DU(a,\ UID)}$ pairs, and uploads details on Blockchain for their own attribute management.

Token generation: GenToken contact will generate a token for the data user to access the file stored on the Cloud server. It collects values from the Blockchain for that file and generates the token. Key K_s for the file was provided from Data Owner encrypts first with the Public Key $P_{U\text{-}DU}$* of the user which is generated from the CAKey contract. Token T_{UID} generated and encrypted with Public Key $P_{U\text{-}DU(a,\ UID)}$ generated from Attribute authority and Encrypted token ET_{UID} sends to the data user.

5) Phase 5: Decryption and access file setup

A data user receives an encrypted token $ET_{\ UID}$ from the Blockchain. After receiving permission to access a file through a smart contract user decrypts the token using the Private Key $P_{R\text{-}DU(a,\ UID)}$ generated from Attribute authority. It decrypts a key K_s for the file using a Private Key $P_{R\text{-}DU}$* of the user which is generated from the CAKey contract. Users can access the file for the time period as per the access rights given by the data owner. Receiving of token $ET_{UID} = (T_{UID,}\ P_{U\text{-}DU(a,\ UID)})$ from contract user computes.

$$E_u = L.\ R^{1/\ P_{R\text{-}DU\ (a,\ UID)}}$$

$$= \prod_x (E_{1,x}\ .\ \left(\frac{e(H(UID),E3,x)}{e(AK_{\rho(x),UID}\ .,E2,x)}\right)^{\frac{1}{SK_{a,UID}}})t_x$$

$$= \prod_x (e(g,g)^{\lambda_x}\ \ e(H(UID),g))^{\frac{c_x}{SK_{a,UID}}})t_x$$

Since $\lambda_{x=} A_x\ v^t = (s, x_1, x_2, \dots, x_n)$ and

$$\sum_x t_x\, A_x . v^t = (1{,}0{,}0, \dots .{,}0).(s, x1, x2, \dots \dots, xn)\ ^t = s$$

Since $E_{x=} A_x\ u^t$ and $u=(0, u_2, u_3, \dots, u_i)$

$$\sum_x t_x\, E_x . u^t = (1{,}0{,}0, \dots .{,}0).(0, u2, u3, \dots . ui)\ ^t = 0$$

$$E_u = P_{R\text{-}DU}\text{*}.\ e(g,g)^s / e(g,g)^s = K_s$$

Finally, with the key user, one can decrypt the file and get the plaintext.

Verification file smart contract will check the integrity of the file by calculating the new hash value and comparing it with the old hash value stored on the Blockchain. If the file verification fails, then Recovery file contact sends a request to the Data owner using OwnerId to replace the original file on a server then the file is accessible to a Data user. SetTimetoAccess contract file assigns time to access the file and when time completes it automatically removes the user from the user list. So further file access is not available to the Data user.

F. Smart Contract Design

We describe our system's smart contract algorithm and interface logic in this section. Smart contracts compiled on solidity and deployed on the Geth Ethereum client. In our scheme, we created a smart contract and deployed it.

There are eleven functions used in a smart contract file.

1) UploadDO (URL, A, H_F, FileId, OwnerId): This function is used by the Data owner to upload the data on the Blockchain which includes the URL of the file, a hash value of the file, file identification number, and owner id of the owner. When the smart contract executes owner's Ethereum address will be retrieved and recorded to the Blockchain.

Algorithm 1. Upload Data Owner Data on Blockchain

```
Procedure UploadDO (URL, A, $H_F$, FileId, OwnerId)
if fileID does not exist then
value[w].URL ≤ URL;
value[w]. A ≤ A;
value[w]. $H_F \leq H_F$;
value[w]. FileId ≤ FileId;
value[w]. OwnerId ≤ OwnerId;
else
return false;
end if
end procedure
```

2) CAKey (w, UID, GP, Userlist): This function is used by a Data user to join the system and after the execution of this contract, the User id will be added to the user list for further communications.

Algorithm 2. Certification Authority Key Generation

```
Procedure CAKey (w, UID, GP, Userlist)
If (UID, A) = True then
Mapping([UID]) ≥ $P_{R\text{-}DU}$*;
Mapping([UID]) ≥ $P_{U\text{-}DU}$*;
Userlist[w] ≥ Userlist[w]U UID;
Count[w]++;
end if
end procedure
```

3) AttributeAuthority (t, authority,n) and AAKey (GP, UID, PR-DU*, w): The Attribute authority function is used by a Data user to join the system, and after execution of this contact User id gets authority to communicate in the system. AAKey function generates the Private Key and Public Key pairs which use in the decryption of tokens which is generated from the Blockchain. Here we use Shamir secret sharing scheme (SSS) to share the secret key between Attribute Authority (AA) and Data User (DU). This scheme is used for secret communication for exchanging data over a network.

Algorithm 3. Attribute Authority

```
Procedure AttributeAuthority (t, authority,n)
If (UID, Userlist) = True then
// Assign authority list
value[t]. authoritylist ≤ authority;
end if
end procedure
```

Algorithm 4. Attribute Authority Key Generation

```
Procedure AAKey (GP, UID, $P_{R\text{-}DU}$*, w)
If (UID, A) = True then
//Generate Public and Private Key pairs for a user using attribute authority
Mapping([UID]) ≥ $P_{R\text{-}DU\ (w,\ UID)}$;
 Mapping([UID]) ≥ $P_{U\text{-}DU\ (w,\ UID)}$;
end if
end procedure
```

4) GenToken (UID, T UID, w) and EncryptToken (UID, ET UID): This function is used when a Data user requests to access the file. After the request receives from the Data user, the contact will check the user ID in the user list if the user is valid, then it'll generate the token. The function collects the values from the Blockchain for the file and generates the token which includes the URL of the file, the hash value of the file, and a key from the Data Owner which is encrypted using a Public Key generated for the user by the Certification Authority key generation contract function. Generated Token encrypted first with the Public Key generated from Attribute Authority key generation contract function. An encrypted token sends to the Data User.

Algorithm 5. Token Generation for a User to Access the File

```
Procedure GenToken (UID, $T_{UID}$,w)
If (UID, A) = True then
//collect values from Blockchain
URL ≤ value[w].URL;
A ≤ value[w]. A;
$H_F$ ≤ value[w]. $H_F$;
$K_s$ ≤ value[w].  $K_s$;
PUK* ≤ Encrypt (PUK, $P_{U\text{-}DU}$*);
//Generate the token
$T_{UID}$ ≤ (URL, A, $H_F$, PUK*, UID);
end if
end procedure
```

Algorithm 6. Encrypt Token Generated for the User to Access File using Public Key of Attribute Authority

```
Procedure EncryptToken (UID, $ET_{UID}$)
If (UID, A) = True then
//Encrypt the token
$ET_{UID}$ ≤ Encrypt ($T_{UID}$, $P_{U\text{-}DU\ (a,\ UID)}$);
end if
end procedure
```

5) DecryptToken (UID, T_{UID}): This function is used by a Data user to decrypt the token received. It also decrypts the key which is used to decrypt a file.

Algorithm 7. Decrypt Token Generated for User to Access File Using Private Key Received from Attribute Authority

```
Procedure DecryptToken (UID, $T_{UID}$)
If (UID, A) = True then
//Decrypt the token
$T_{UID}$ ≤ Decrypt ($ET_{UID}$, $P_{R\text{-}DU\ (a,\ UID)}$)
//Decrypt the Key received in the Token
PUK ≤ Decrypt (PUK*, $P_{R\text{-}DU}$*)
end if
end procedure
```

6) Verificationfile (FileId, URL, UID), Recoveryfile (FileId, URL, UID), and Update (OwnerId, FileId, URL): Verificationfile function verifies the integrity of the file. It computes a new hash value and compares them with the old hash value stored on the Blockchain. The value which is stored on a blockchain is immutable that can't be altered, modified, or deleted. If the file is incorrect then it calls the Recoveryfile function and informs the Data owner to replace the original file on provided URL. The Update function will replace the new file and message to the Data user that you can download the file.

Algorithm 8. Integrity Verification of the File

```
Procedure Verificationfile (FileId, URL, UID)
//Generate a New Hash of the fie
$NH_F$ =$GEN_h$(FileId);
//compare the new hash with the hash value stored on the Blockchain
If ($NH_F$, $H_F$) =True then
Message (UID," file is correct");
else
Message (UID," file is Incorrect");
call Recoveryfile (FileId, URL, UID)
end if
end procedure
```

Algorithm 9. Recovery of the File

```
Procedure Recoveryfile (FileId, URL, UID)
// Get OwnerId of the file from Blockchain
OwnerId= value[w]. OwnerId;
Message (OwnerId," Update File (FileId, URL)");
Recivemessage ("file updated (OwnerId, FileId, URL)");
Message (UID," File updated (FileId, URL)");
end procedure
```

Algorithm 10. Data Owner File Update

```
Procedure Update (OwnerId, FileId, URL)
// replace new file with old file using URL.
Updatefile (OwnerId, FileId);
Message (SCId," File updated (FileId, URL)");
end procedure
```

7) SetTimetoAccess (UID, ATIME, URL, $P_{U\text{-}DU}$*): This function will set the time period to access the file or data to the data user. When access time ends it automatically removes a user from the user list.

Algorithm 11. Set the Time to Access the Data as per Access Policy

```
Procedure SetTimetoAccess (UID, ATIME, URL, $P_{U-DU}$*)
If (UID, A) = True then
// Assign a Time period to access the data
C=0;
C++;
If (C, TIME) = True then
Remove (Userlist, UID);
end if
end procedure
```

V. Implementation, Security, Functional, Experimental, and Performance analysis

A. Implementation Details

Experiments were carried out using the system configuration like windows 11,16 GB RAM and 2.7 GHz intel i7 11 generation processor. Implementation has been done using Node.js is used for server-side programming and backend API services. An Interplanetary File System (IPFS) is used to upload the file to the cloud storage server. Ganache is the Ethereum Blockchain that is used to run and test contract files. Truffle is used to build, debug and deploy smart contracts over the Ethereum Blockchain network. Visual studio code is used as an editor to write a smart contract and other files. The used technology details are mentioned in Table II.

TABLE II: Technology Details

Technology	Version
Node.js	16.15.0
Ganache	2.5.4
Truffle	5.6.4
Visual Studio Code	1.72.0

B. Security Analysis and Discussion

In this paper, we have implemented an access control system with integrity verification and recovery. In this section, we give a security analysis of the CP-ABE scheme over a not fully trusted Cloud environment. we testified our proposed system meets confidentiality, integrity, security against attack, and auditable access control.

C. Confidentiality

A Cloud server is only storing the encrypted file. It is not responsible for any Private Key and Public Key pairs. All the functions are done by a smart contract. We are using the attribute authority to communicate with the system using a smart contract so there is no third party. Tokens are only generated with a particular user using attribute keys. It is hard for anyone to decrypt data without the private and Public Keys of the user. These tokens are only available to the person who is a valid user in the user list and the certificate authority has provided keys to communicate with the system.

Theorem 1: Our proposed system model can be indistinguishable under adaptive chosen-ciphertext attacks (IND-CCA) If there is no polynomial-time attack on the system model with a significant probability advantage.

Proof: The proof method from the scheme [27], We demonstrate that the confidentiality of our system satisfies security under a chosen-ciphertext attack using the proof methodology in Scheme.

The following game is played between adversary A and challenger B.

Initialization phase: B executes (1λ) to generate the system private key PR = (g, (s, x_1, x_2, …, x_n)) and the system public key PU = (EX, Tu, [(0, u_2, u_3, …, u_i))] ^ t). Then, B sends PR to A and creates a set R and list L that are both originally empty.

Phase 1 of the inquiry: A initiates the following two inquiries to B.

1) Private key inquiry: A adaptively asks B for the attribute set x private key; B uses key generation and returns P_R = (s_0, x_1, x_2, …, x_n) back to A.B assigns $x \cap R \geq R$.
2) Transformation key inquiry: After receiving the token inquiry request from A, B looks for (x, P_U, $tP_{U)}$ in the list L. B returns tP_U to A if exists (x, P_U,tP_U); if not, B picks a random number $y \in Z$ and does calculations (x, P_U, $tP_{U)}$. B then adds x to tP_U and to the list L before giving it back to A.

Challenge phase: A sends B access structure W_α, W_β, and messages M_α, M_β of identical length. B chooses $\mu \in \{0,1\}$ and runs ciphertext generator $E_u = P_R^* \cdot e(g,g)^s / e(g,g)^s$ and starts it. B then sends E_u to A.

Phase 2 of the inquiry: A request for the queries again. B answers the queries similar to phase 1.

1) Key generation query: A ask for the private key P_R*. B chooses random number $r \in Z$. It executes the key generation algorithm and sends generated key P_R* tp A.B selects a random number to construct a private key and generates key and shares to A.
2) Challenge phase: B sends private key P_R* and chooses random number r and runs key generation algorithm and returns n $P_R^* \cdot e(g,g)^s$ private key.

Guess: The output $\mu' \in \{0,1\}$ guess by A. if $\mu' \neq \mu$ then the attack is successful. According to Scheme [27] proof of Definition [5], it is challenging for A to guess μ' and μ was chosen at random during the encryption function. We demonstrate that our scheme's confidentiality satisfies security requirements against a chosen-ciphertext attack.

D. Integrity

A Cloud server can be attacked by some malicious users and files stored on it can be affected or changed. In our proposed system, we have the ability to check whether a data user is using the correct file or not. Only the owner of the file can replace the data on the Cloud so there is less chance of any fraud activity. Integrity verification of the file is also done at the time of accessing the file using a smart contract so there is no computational overhead. The data access user replaces the key allocation function after receiving the semi-decrypted ciphertext $E_{x=} A_x\ u^t$ from the Cloud server and calculates the public key using the blinding factor ρ. When an equation is established, a smart contract checks the equation and produces the results using key $PK_{a,UID} \cdot e(g,g)^s$. The data access user then keeps on decrypting partially decrypted ciphertext. Otherwise, the decryption is completed, and the smart contract outputs 0.

After the decryption process hash value is generated using the hash function NFHASH and the old hash value FHASH reads from Blockchain compares and if the results Boolean value is 0 then the integrity of the file is verified.

E. Security Against Attack

In the proposed system, Cloud server is storing encrypted file which is uploaded by a data owner. There is no other information on file like Public Keys stored on it. So, anyone can't directly get access od data from a Cloud server. All over control is with a data owner. In the proposed system, there are only users who can access data on the Cloud who are registered with the certification authority. A token is generated using attribute authority for a specific user with a user id. So, it is impossible to get keys $e(g,g)^{P_{R-DU_{\alpha\mu,i}}}$, $g^{P_{R-DU_{\beta\,\mu,I}}}$ generated from attribute authority to interact with the system without a user id UID and other attributes to interact with the system. A Key is also provided in encrypted form using a Public Key $P_{U\text{-}DU}$* generated by the user. Factor $e(g,g)^s$ can't be recovered by the attacker. So, it is impossible for an attacker to get all the keys and manage to hack the data.

F. Auditable Access Control

In our proposed scheme all logs are recorded on the Blockchain in a trustable and immutable environment. A data owner and data user can verify if the access policies have been evaluated and how each request is processed. Therefore, we can say that our proposed system inherits the property auditability of Blockchain.

G. Discussion

In our proposed scheme, we have used the CP-ABE scheme with a public blockchain. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme uses a blinded access policy to share the owner's data with authorized data users. Ciphertext-policy Attribute-based encryption scheme provides dynamic management of access control with blockchain. This scheme protects the privacy of data in a communication network. It provides a flexible and fine-grained access control mechanism. It ensures the invisibility of users because encrypts data with the user's private attributes. It makes the encryption and decryption process faster because it reduces the number of keys.

Blockchain provides immutable data with high-security features. We have used a Public blockchain that works on a decentralized solution and protects it from single-point failure. Public blockchain can be secured with automatic validation methods and encryption. It is a permission-less blockchain, where there is no central authority, the data stored on the blockchain can't be changed once it will validate. Public blockchain provides high security and is open to everyone. There are no specific rules and limitations to using a public blockchain. A public blockchain is truly decentralized and transparent. Every user has a copy of the ledger and maintains every activity record. The block added to the chain can't modify, deleted, or altered by anyone so it provides immutability. The user has complete power to work with the system without the control of anybody. Cryptographic hash functions are used to build the block.SHA-256 algorithm is used to create hash values. Various consensus algorithms are used to add a block to a chain. After successfully passing from critical mathematical operations the block will be added to the blockchain.

H. Functional Analysis

The functional comparison with other systems is shown in Table III.

TABLE III: FUNCTIONAL ANALYSIS AND COMPARISON [12, 23, 24, 32]

Literature	Central Authority	Blockchain	Access log	Integrity
[12]	×	√	√	×
[23]	√	√	√	×
[24]	×	√	×	×
[29]	×	×	√	×
[30]	×	×	√	×
[32]	√	√	×	×
Proposed System	×	√	√	√

The performance of the proposed scheme is compared with another scheme by a central authority, based on Blockchain, based on access logs, or whether the integrity of the file is detected or not. Sukhodolskiy and Zapechnikov [12] proposed an access control system that transfers hash codes through a Blockchain ledger. Qin and Huang *et al.* [23] shared all ciphertext data over the Blockchain with an access time period. It does not have the facility to trace the access log and verification data. Wang *et al.* [24] proposed a multi-authority attribute-based scheme that provides attribute-based encryption and accessing structure on the Blockchain. Benil and Jasper [32] proposed a health system that uses central authority management using Blockchain technology. Xue *et al.* [29] proposed an access control system for collaborative access control for cloud storage without blockchain technology. Hao and Huang *et al.* [30] implemented IoT based access control system which uses an attribute hiding policy. It hides the data user's identity in an access control mechanism.

In our proposed scheme, we use a multi-authority CP-ABE scheme for the access control mechanism and remove the concept of central authority. Due to the Blockchain technology, we provide immutable data and access logs can verify easily. This system also manages the integrity of the data. Therefore, our solution provides more usability and applicability.

I. Experimental Analysis

The proposed model is compared to the framework which uses the attribute-based encryption scheme. We have deployed the smart contract using Ganache and analyzed the execution of every algorithm computation with the gas computation method in a smart contract. Table IV lists the algorithms which use gas cost and the cost of the experiment. Here we set 1 ether to 200USD, 1 gas price to 1 Gwei, where 1 Gwei= 10^{-9} ether=10^9 Gwei.

TABLE IV: SMART CONTRACT COST (1 ETHER =200 USD, 1 GAS PRICE = 1 GWEI)

Function	Gas Used	Cost (Ether)
UploadDO	1271298	0.002535357
CAKey	911923	0.001819735
AttributeAuthority	69500	0.000136993
AAKey	110257	0.000221537
GenToken	40088	0.000082398
EncryptToken	21354	0.000042699
Verificationfile	24798	0.000048967
Recoveryfile	25023	0.000051387
Update	28871	0.000057854
SetTimetoAccess	35846	0.000065367

J. Performance Analysis

In this section, we give an experimental analysis of our system. We conduct simulations to evaluate performance with BMAC with our proposed scheme. All experiments were carried out under the system windows 11,16 GB RAM and 2.7 GHz intel i7 11 generation processor. See Table V for Communication Cost.

TABLE V: COMMUNICATION COST [23, 25, 26]

Method	System Initialization			Key/Token Distribution	
	CA	Each AA	User	Each AA	User
[23]	$3\|G_1\|+\|SC\|$	$(2NA-1)(\|G_1\|+\|G_2\|)N_{attr}$	$\|G_1\|$	$N_{attr}\|G_2\|+2\|G_1\|$	$3\|G_2\|$
[25]	NA	$(NA+NA_n)(\|G_1\|+n\|G_2\|)N_{attr}$	$\|G_1\|$	N_{attr} $2\|G_2\|+\|G_1\|$	$3\|G_2\|$
[26]	NA	$(2NA-1)\|G_1\|(N_{attr}+\|G_2\|)$	$\|G_1\|$	N_{attr} $2\|G_2\|+2\|G_1\|$	$3\|G_2\|$
Our Method	$\|G_1\|+\|SC\|$	$(2NA-1)(\|G_1\|+\|G_2\|)N_{attr}$	$\|G_1\|$	$N_{attr}\|G_2\|+2\|G_1\|$	$\|G_2\|$

K. Communication Overhead

Let |G1| and |G2| be the size of elements in G. |SC| is the size of smart contracts and Nattr is the average size of the attributes from AAs. A comparison between BMAC [23, 25, 26] and our proposed scheme is shown in Table V. The certification authority uploads the parameters on Blockchain and deploys smart contracts, so the communication overhead is |G1|+|SC| and in BMAC it is 3|G1|+|SC|. Token size is equal to |G2| whereas in BMAC [23] it is 3|G2|. Attribute registration and initialization phase in attribute authority are the same in both systems. In BMAC [23], central authority manages attribute distribution wherein our scheme automatically smart contract will manage attribute distribution, so it has less communication overhead. Yang and Chen *et al.* [25] and Li [26] proposed the systems that there is no certification authority, so it will have an overburden of user management and attribute management in a single system.

L. Computation Overhead

Attribute Authority setup algorithms and token generation algorithms are jointly managed by all authorities. The average execution time in both algorithms grows linearly as the number of attributes increases from 1 to 20. Fig. 4 and Fig. 5 show the computation overhead for these two algorithms with the number of attributes.

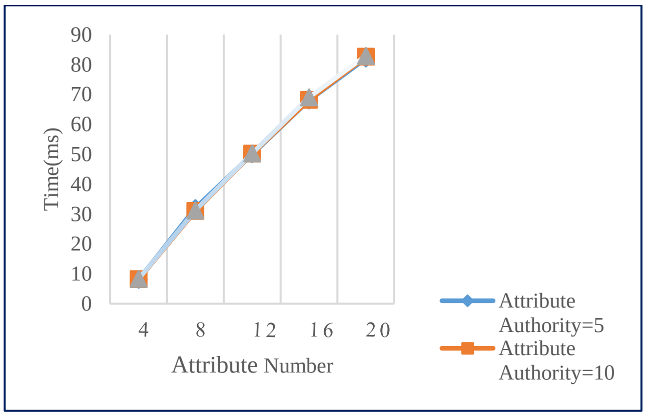

Fig. 4. The average execution time of the Attribute Authority setup algorithms.

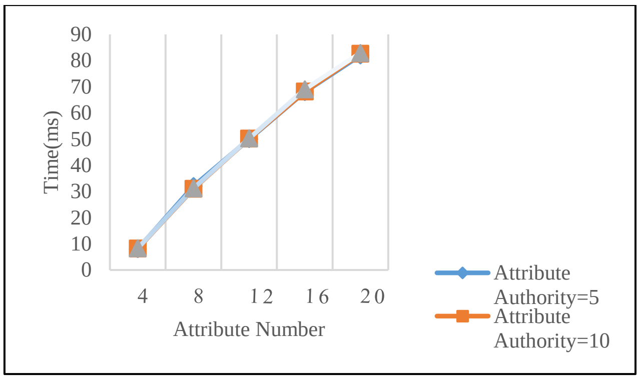

Fig. 5. The average execution time of the Token generation algorithm.

VI. SOME COMMON MISTAKES

Figs. 6 shows the average execution time for encryption and decryption of tokens. We have compared it with the existing system of [25, 26]. In our proposed method Blockchain and smart contract, methods perform more tasks so encryption and decryption of token algorithm execution time are basically constant. A data owner encrypts the file and uploads the file to the server other tasks are handled by a smart contract. The encrypted token is generated by Blockchain and smart contract. So, in this case there is less interference from the owner. Only the user has to download the data and access the data as per the time provided by the data owner.

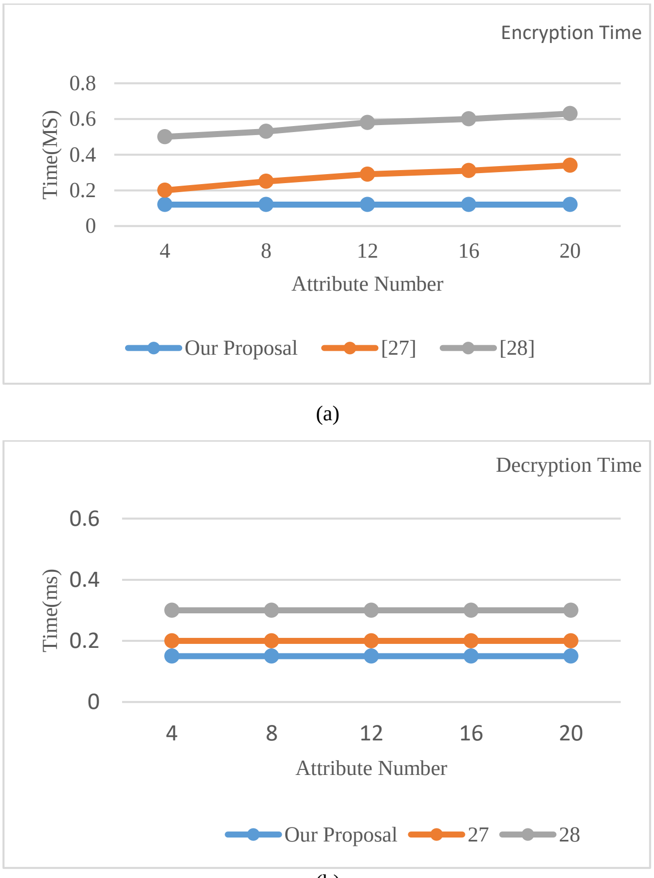

Fig. 6. (a) The average execution time of the Encryption algorithm. (b) The average execution time of the Decryption algorithm.

VII. Conclusion

In this paper, we present a system model which provides security to the data stored on the Cloud using Blockchain Technology. By using Blockchain technology, we can provide the data that is immutable and tamper-proof. All transactions of access or managing to files, rejection, and the inability to edit any data is guaranteed through the use of the Blockchain and smart contracts in our system. In order to maintain security without a third party, we use a smart contract that automatically executes when the condition matches. The Cloud server does not have any information about keys. All the transactions related to data stored in log tables and logs are maintained using Blockchain. The present scheme has also provided management of recovery files. Moreover, this scheme accomplishes ownership management. Our system required less computational and communication overhead and offers more security to data and logs, integrity verification, ownership, and dynamic access control.

Conflict of Interest

The authors declare that they have no conflicts of interest.

Author Contributions

Parin Patel: research for related works, analysis, design, amelioration of the proposed model and drafting of the article. Hiren Patel: total supervision of paperwork, review, comments, assessment, etc. All authors contributed to the article and approved the final version.

References

[1] Project Pro. (2023). How big data analysis helped increase Walmarts Sales turnover? [Online]. Available: https://www.dezyre.com/article/how-big-data-analysis-helped-increase-walmarts-sales-turnover/109

[2] S. Nakamoto. (2008). Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] M. Vukolič, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Proc. International Workshop on Open Problems in Network Security*, Springer, 2015, pp. 112–125.

[4] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[5] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 2567–2572.

[6] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, O'Reilly Media, Inc., 2014.

[7] M. Pilkington, "Blockchain technology: Principles and applications," in *Research Handbook on Digital Transformations*, 2016, p. 225.

[8] C. Dannen, *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*, Apress, 2017.

[9] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[10] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," Work Paper, 2016.

[11] J. Li, J. Wu, and L. Chen, "Block-secure: Blockchain based scheme for secure P2P cloud storage," *Information Sciences*, 2018, doi: 10.1016/j.ins.2018.06.071

[12] I. Sukhodolskiy and S. Zapechnikov, "A blockchain-based access control system for cloud storage," in *Proc. 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2018.

[13] S. Ramamoorthy and B. Baranidharan, "CloudBC—A secure cloud data access management system," in *Proc. 2019 3rd International Conference on Computing and Communications Technologies (ICCCT)*, 2019, doi: 10.1109/iccct2.2019.8824828

[14] J. Xue, C. Xu, Y. Zhang, and L. Bai, "DStore: A distributed cloud storage system based on smart contracts and blockchain," in *Proc. 18th International Conference, ICA3PP 2018*, Guangzhou, China, November 15–17, 2018, doi: 10.1007/978-3-030-05057-3_30

[15] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 2018 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2018.

[16] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in P2P cloud storage," in *Proc. 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Singapore, 2018, pp. 561–568, doi: 10.1109/PADSW.2018.8644863

[17] P. C. Wei, D. Wang, Y. Zhao, S. K. S. Tyagi, and N. Kumar, "Blockchain data-based cloud data integrity protection mechanism," *Future Generation Computer Systems*, vol. 102, pp. 902–911, 2020, https://doi.org/10.1016/j.future.2019.09.028

[18] A. Omar, M. Bhuiyan, A. Basu, S. Kiyomoto, and S. Rahman, "Privacy-friendly platform for healthcare data in cloud based on blockchain environment," *Future Generation Computer Systems*, vol. 95, pp. 511–521, 2019, https://doi.org/10.1016/j.future.2018.12.044

[19] J. Li, J. Wu, L. Chen, and J. Li, "Deduplication with blockchain for secure cloud storage," in *Proc. 6th CCF Conference on Big Data*, Xi'an, China, October 11–13, 2018, doi: 10.1007/978-981-13-2922-7_36

[20] M. Kumar, A. K. Singh, and T. V. S. Kumar, "Secure log storage using blockchain and cloud infrastructure," in *Proc. 9th ICCCNT 2018*, Bengaluru, India, IEEE, 2018.

[21] W. Pourmajidi and A. Miranskyy, "Logchain: Blockchain-assisted log storage," in *Proc. 2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018.

[22] A. Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in *Proc. ISWC 2017: Lecture Notes in Computer Science*, Springer, Cham., 2017, vol. 10587.

[23] X. Qin, Y. Huang, Z. Yang, and X. Li, "A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing," *Journal of Systems Architecture*, vol. 112, 101854, 2021, https://doi.org/10.1016/j.sysarc.2020.101854

[24] S. Wang, X. Wang, and Y. Zhang, "A secure cloud storage framework with access control based on blockchain," *IEEE Access*, vol. 7, pp. 112713–112725, 2019, doi: 10.1109/ACCESS.2019.2929205

[25] X. Yang, A. Chen, Z. Wang, and S. Li, "Cloud storage data access control scheme based on blockchain and attribute-based encryption," *Computational Security and Communication Networks, Hindawi*, 2022, https://doi.org/10.1155/2022/2204832

[26] X. Li, "A blockchain-based verifiable user data access control policy for secured cloud data storage," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–12, Apr. 2022, https://doi.org/10.1155/2022/2254411

[27] Y. Zhao, X. Zhang, X. Xie, and S. Kumar, "A verifiable hidden policy CP-ABE with decryption testing scheme and its application in VANET," *Transactions on Emerging Telecommunications Technologies*, p. e3785, 2019.

[28] F. Sammy and S. M. C. Vigila, "An efficient blockchain based data access with modified hierarchical attribute access structure with CP-ABE using ECC scheme for patient health record," *Security and Communication Networks*, vol. 2022, pp. 1–11, Mar. 2022, https://doi.org/10.1155/2022/8685273

[29] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. L. Wei, and P. Hong, "An attribute-based controlled collaborative access control scheme for public cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 2927–2942, Nov. 2019, doi: 10.1109/TIFS.2019.2911166

[30] J. Hao, C. Huang, 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, Apr. 2019, https://doi.org/10.1016/j.comnet.2019.02.008

[31] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2011, vol. 6632, pp. 568–588, http://doi.org/10.1007/978-3-540-70936-7_28S

[32] T. Benil and J. Jasper, "Cloud-based security on outsourcing using blockchain in e-health systems," *Computer Networks*, vol. 178, 107344, 2020, https://doi.org/10.1016/j.comnet.2020.107344

[33] H. Wang *et al.*, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Inf. Sci.*, vol. 519, pp. 348–362, 2020.

[34] P. Sharma, R. Jindal, and M. Borah, "A blockchain-based cloud storage system with CP-ABE-based access control and revocation process," *The Journal of Supercomputing*, vol. 78, 2022, doi: 10.1007/s11227-021-04179-4