IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 15, 2022, accepted March 26, 2022, date of publication April 1, 2022, date of current version April 11, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3164081

Towards Blockchain-Based Secure Storage and Trusted Data Sharing Scheme for IoT Environment

ZIA ULLAH[1], BASIT RAZA[1], HABIB SHAH[2], SHAHZAD KHAN[3], AND ABDUL WAHEED[4,5]

[1]Department of Computer Science, COMSATS University Islamabad, Islamabad 44000, Pakistan
[2]Department of Computer Science, College of Computer Science, King Khalid University, Abha 62529, Saudi Arabia
[3]Department of Information Security, Military College of Signals (MCS), NUST, Islamabad 44000, Pakistan
[4]Department of Computer Science, Northern University, Nowshera 24100, Pakistan
[5]School of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea

Corresponding author: Basit Raza (basit.raza@comsats.edu.pk)

This work was supported by the Deanship of Scientific Research, King Khalid University, Saudi Arabia, under Grant RGP.2/190/42.

ABSTRACT Nowadays, cloud-based storage systems play a vital role in IoT data storage, processing, and sharing. Despite its contribution, the current cloud-based architecture may cause severe data leakage or jeopardize user privacy. Meanwhile, the cloud-based architecture heavily relies on a trusted third-party auditor (TPA) and runs in a centralized control manner. However, the TPA may not be a completely trustworthy entity, and a single point of failure might cause the centralized system to collapse. Fortunately, with the advent of blockchain technology, the decentralized storage model has gained popularity. A decentralized storage system successfully eradicates the rule of TPA, solves the problem of a single point of failure, and has many advantages over a centralized control architecture, such as low storage prices and high throughput. This study offers a blockchain-based decentralized distributed storage and sharing scheme that provides end-to-end encryption and fine-grained access control. In our proposed IoTChain model, fine-grained permission is based on attribute-based access control (A-BAC) policy by employing the Ethereum blockchain as an auditable access control layer. Smart contracts are tailored for the IoTChain model, which combines the Ethereum blockchain and the interplanetary file system (IPFS). We used an advanced encryption standard (AES) for encryption and the elliptic curve Diffie-Hellman key exchange protocol for secret key sharing between data owners and users. Also, the proof-of-work (PoW) consensus mechanism is replaced with a proof-of-authority (PoA) to minimize system transaction cost and boost system throughput. Additionally, our solution has been tested on the Ethereum official test network Rinkeby, and the results demonstrate that our approach is realistic and economical on the IoT data.

INDEX TERMS Access control, data encryption, data storage via blockchain, Ethereum blockchain, Internet of Things (IoT), IPFS, smart contract.

I. INTRODUCTION

With the tremendous advancement of internet technologies, there is an exponential growth in the Internet of Things (IoT). Due to its broad application, it has been extensively adopted in military surveillance [1], e-smart health [2], traffic monitoring [3], industrial control [4], and so on. IoT devices employ various technologies, including sensing, computation, and wireless connectivity, to generate a large data stream. It not only improves living standards but also contributes to the world economy. It is predicted that the IoT will connect 30 billion devices and create about $7.1 trillion in the world economy by 2025 [5]. IoT devices are data-centric in which data generation exponentially grows [6]. Therefore, storing such data on a cloud-based storage system arises various issues, as shown in Fig. 1. Such a centralized approach is vulnerable to a vast number of security and privacy issues, such as single-point failure, false data injection, vulnerability to Sybil attack, trust issues among participants, and problems in file access and retrieval operations [5]– [7].

IoT devices collect information from various monitoring areas where they are deployed, as shown in Fig. 1. They send the data to the Cloud Service Providers (CSP), a traditional

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

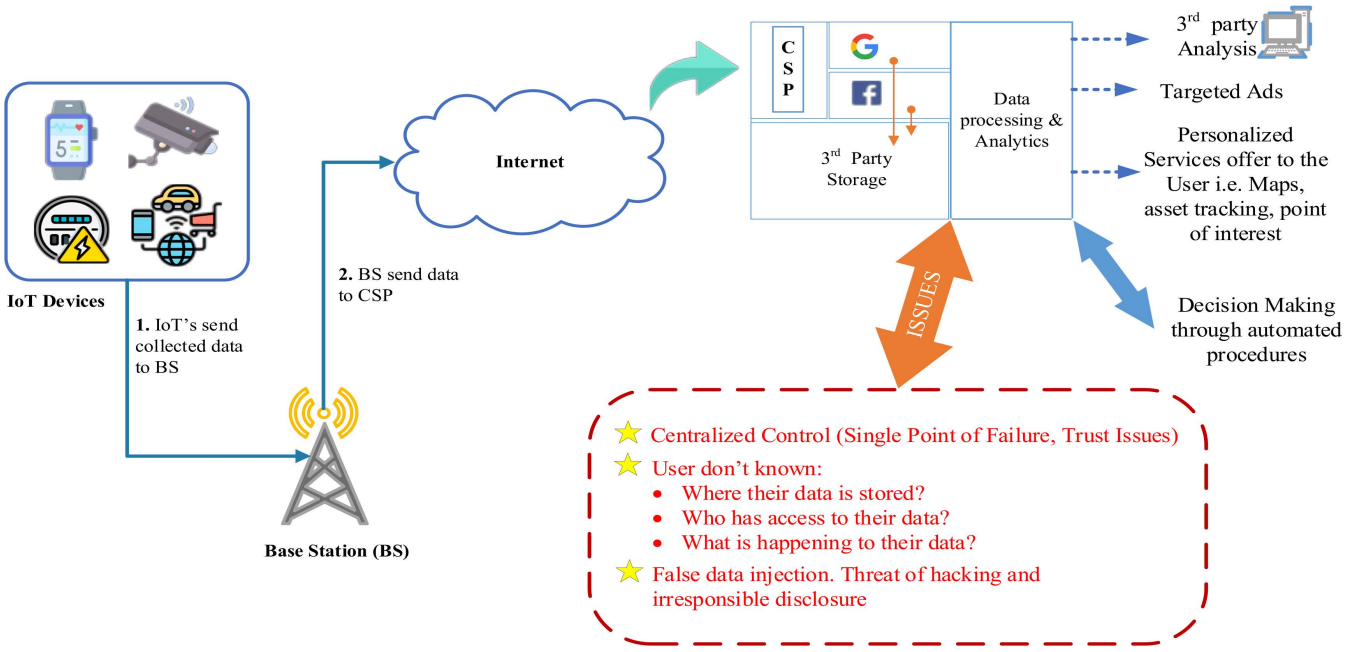

FIGURE 1. IoT data storage issues with traditional cloud service providers (CSP).

cloud-based storage system. These service providers mostly follow a centralized control approach where trusted third parties store the data. Although such a centralized system may look effective from the outside, it has numerous issues, such as maintaining such a system is costly and easily hackable, which will cause catastrophic consequences of a single point of failure [8]. Even if the system is backed up, CSP may still suffer significant Force Majeure (owners will be unable to access their data). Furthermore, a user has trust issues with the third party, and they do not know where their data is stored and what is happening to it. Who has access to it, and is there any unauthorized disclosure to third parties? Research shows that a bug in Google Plus resulted in approximately 600,000 user information leakages and is an example of one of the CSP vulnerabilities [5].

Therefore, the future needs a decentralized storage mechanism that significantly improves efficiency, data transparency and provides trust among the participants without the involvement of third parties. Fortunately, with the advent of bitcoin [9], its underlying technologies, i.e., Ethereum blockchain combined with the interplanetary file system (IPFS), will provide an efficient solution to a distributed storage system. Ethereum is a permission-less and public blockchain. Smart contracts, which are self-executing activities recorded on the Ethereum blockchain and used to develop dApps, are the key enablers for numerous innovations. A decentralized approach is a solution to protect better privacy and data availability, in which data is stored independently on different nodes in the network. Moreover, the distributed system significantly eliminates the problem of a single point of failure. It also lowers the price of data storage compared to the traditional cloud or third-party storage. It works similarly to the standard internet but is different in features, wherein the data is accessed by content-based addressing instead of location-based addressing. The key contributions to this paper are as follow:

1) We eliminate the traditional cloud or third-party storage problems by storing the IoT data on a decentralized storage system known as IPFS, which combines with the Ethereum blockchain.
2) *DO* set an access control policy so that unauthorized personnel cannot control or view data. We introduce attribute-based access control (A-BAC) and AES-128 encryption schemes for better data security and privacy that encrypt the IoT stream before uploading to IPFS.
3) Furthermore, the encrypted hashes are stored in the Ethereum smart contract. Moreover, the elliptic curve Diffie-Hellman key exchange protocol is used to securely distribute the secret key that solves the problem of key management; in our solution, a trustworthy PKG (private key generator) is not required. Whenever a data user forgets his private key, he can only access the transaction details from the Ethereum blockchain and get the private key.
4) IoTChain is an incentive-based approach. The nodes that store the data will be rewarded with digital currency. Filecoin, a digital currency introduced by IPFS, will be rewarded as an incentive to encourage the storage nodes. Furthermore, smart contracts are deployed on the Ethereum blockchain to implement encrypted keyword searches in the IPFS.
5) In the IoTChain model, the proof-of-work (PoW) consensus mechanism is replaced with a proof-of-authority (PoA) to minimize transaction cost and boost system

throughput. Additionally, the smart contracts will operate in good faith and as per their logic.

6) We simulated our scheme via Ethereum official test network Rinkeby, and the corresponding performance and transaction cost were analyzed.

II. BACKGROUND AND RELATED WORK

A. BLOCKCHAIN OVERVIEW

In the recent era, blockchain cryptocurrency (such as BitCoin [9], Ethereum [10], ZCash [11] etc.) has become a hot and emerging technology that has attracted more and more attention from industries and researchers. Satoshi Nakamoto firstly introduced the blockchain concept in a cryptocurrency [9]. In addition, it is a disruptive technology in many non-financial applications, such as decentralized storage systems [12], decentralized internet of things (IoT) [13], Vehicular Ad-hoc Networks (VANETS) [14], identity management [15], and public utilities [16], and so-on.

These use cases aim to take advantage of blockchain essential features, such as decentralized control, immutable and distributed properties, cryptographic security, robustness, and capability to run smart contracts. Blockchain technology is essentially a decentralized, distributed ledger system that records all the transactions on a peer-to-peer computer network [9], as shown in Fig. 2. Each participant in the chain holds an identical copy of the transaction record, and every time a new transaction occurs, distributed ledger technology DLT) adds the records to every participant ledger. In this way, a tamper-proof record of transactions with a cryptographic signature called a hash is stored in a series of linked blocks.

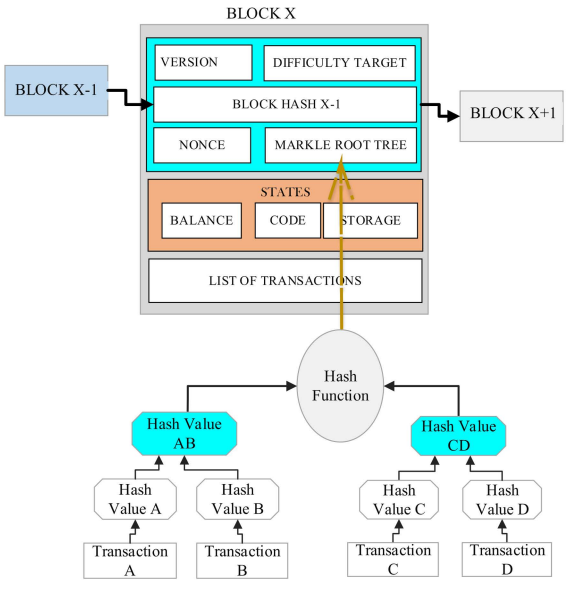

FIGURE 2. Structure of blockchain network.

1) ETHEREUM BLOCKCHAIN

In 2013, Ethereum [10] came into the world as an open-source, programmable blockchain with a turing-complete scripting language that runs on Ethereum virtual machine (EVM). The Ethereum network is more than a payment system that allows writing smart contracts and building decentralized applications (dApps), making a more sophisticated blockchain network [17]. Like Bitcoin, it is used only for digital payment systems in which transactions are performed by the simple logic of a stack-based turing scripting language. While the Ethereum blockchain is powerful enough to run and implement any program defined with similar computational speed. If Bitcoin represents digital money, likewise, Ethereum represents programmable money. Both Bitcoin and Ethereum work on the proof-of-work (PoW) consensus algorithm [18], while Ethereum plans to move to the proof-of-stake (PoS) algorithm by 2022 for scalability and a more user-friendly approach, which is the most significant update in Ethereum history, known as Ethereum 2.0. There is no fraud, downtime, censorship, or third-party involvement in Ethereum blockchain [19]. The main components of the Ethereum blockchain are as follows:

TABLE 1. Notations table.

Notation	Description
DO	Data Owner
DU	Data User
$\hat{e}$	Bilinear Mapping
$\mathcal{P}$	Access Control policy
$\mathcal{S}$	User Attribute set
PK	Public Parameter
DO_{MK}	Data owner master key
DU_{SK}	Data user secret key
λ	System security parameter
CT_F	Ciphertext file
CT_K	Encrypted ciphertext key
CT_l	Encrypted ciphertext location
F_K	File key
Kw	Keyword
$H_{Location}$	File location on IPFS
Enc_K	AES encryption key
Dec_K	AES decryption key

2) ETHEREUM BLOCK CONFIRMATION

Block confirmation time and transaction speed on the Ethereum network are faster than the Bitcoin network. This platform takes 10-20 seconds in terms of block confirmation time [17]. While in Bitcoin, it takes 10 minutes to confirm and validate each block. Similarly, from the statistics on transaction speed, Ethereum has suppressed Bitcoin, in which 5-7 transactions per second are considered within the margin. Ethereum 2.0, with its PoS consensus algorithm, is expected to handle 100,000 transactions per second.

3) ETHEREUM ACCOUNT

There are two different types of Ethereum accounts [20]: externally owned accounts (E.O.A) and contract accounts;

a 20-byte alphanumeric id represents both such as: 0*xd*96*dfe*18*b*6*daf* 5*ec*36*d*15*a*0*e*7*a*61811*afd*4*f* 1600. An external user's private key controls an E.O.A. It has an ether balance, sends a transaction, is controlled by a private key, and has no associated code. The contract account has an associated code, corresponding balance, and nonce. Furthermore, it is managed by the code recorded in the account and is activated whenever it gets ether from E.O.A. The contract account cannot send a transaction on its own; however, the transfer originates from E.O.A.

4) ETHEREUM TRANSACTION

A transaction in the Ethereum platform executes on a call of an associated code to send a signed data message from one account to another Ethereum account. To generate a signature on a transaction, the sender's secret key is used to sign it. A sender's signature is mandatory before submitting the transaction to the network. A transaction contains the recipient's information, account nonce, amount transferred, the smart contract byte code, the transaction fee, known as ''Gas'' or ''Gas Limit'' and the sender's signature. Moreover, a transaction can also be used to publish smart contract code on the Ethereum blockchain. In our proposed IoTChain model, we embed ciphertext of user attributes set and file location and store them on the Ethereum blockchain, making them immutable records. These attributes are compiled in JSON format before being encoded in alphanumeric code. The *eth_getTransactionReceipt* method, provided by Ethereum officials, will return the newly created smart contract once uploaded to the Ethereum blockchain through the JSON API interface.

5) ETHEREUM TRANSA

Gas is a key building block of the Ethereum blockchain that measures the amount of processing effort necessary to implement a particular operation. The token used for the transaction fee is Ether (ETH). If a person executes a transaction operation on EVM, from one account to another Ethereum account, or a complex state-changing operation through a smart contract, the sender has to pay the network validator (minor node) as a transaction fee, which is measured in gas and gas limit. The transaction fee is paid in ETH [10] or in a smaller denomination called Gwei [1 ETH = 1,000,000,000 Gwei (10^9)]. Gas prices are paid in the native currency. Ether serves two purposes. First, it prevents the network from being a bad actor by executing unnecessary transactions that cause congestion in the entire network. Second, it acts as an incentive for network validators (each minor node receives, broadcasts, and verifies every transaction in the Ethereum network). The minor nodes can verify and add blocks to the listed gas limit. If the total amount of gas price is less than or equal to the stated gas limit, the transaction happens; otherwise, a minor cannot verify a transaction [20]. In this way, the minor nodes verify the transactions listed and keep their ledgers synchronized.

6) ETHEREUM CONSENSUS MECHANISM

A distributed consensus method on the Ethereum network defines which blocks can be approved and added to the ledger. The minors use a modified version of the proof-of-work (PoW) consensus algorithm called Ethash (modified Dagger-Hashimoto algorithm) [18]. The Ethereum blockchain is based on a PoW consensus mechanism, requiring minor nodes to compete by solving a complex cryptographic puzzle by repeatedly building blocks with random numbers until the correct number is found. The minor nodes ensure three properties by solving this puzzle: a minor has to invest corresponding computational power to complete the puzzle; the mining process is entirely random, and any other peer node can easily verify a successful minor's claim. If all goes well and the verification procedure is completed, the new block is permanently signed onto the blockchain, and the database is updated successfully. Therefore, the mining process has a significant impact on the security of the Ethereum blockchain.

In the worst-case scenario, a malicious node injects false transaction record blocks into the chain. As a result, the peer nodes adopt an implicit consensus method as a further step. The peer node can check the newly created block, and if an anomaly is identified, such as a discrepancy in the linked hash value, incorrect transaction verification, or ownership, the peer nodes maintain the blockchain's initial state despite accepting a new block [19].

B. HYPERLEDGER BLOCKCHAIN

The most popular permission and private blockchain is Hyperledger [21] and is supported by the Linux Foundation. Hyperledger blockchain limit the number of peers who can access them. In contrast to a permission-less network, everyone can contribute to the canonical chain. A proof-of-work (PoW) consensus is used in Bitcoin and Ethereum, both permission-less blockchains. Whereas, the Ethereum blockchain is permissionless and public. The comparisons among different distributed ledger technology (DLTs) are shown in Table 2. In private settings, node identities are known to all, so most blockchains rely on one of the familiar protocols of distributed consensus. The PBFT [22] protocol is an active protocol that is in use today. Besides deterministic consensus, another key property of private blockchains is that they support smart contracts which can express highly complex transaction logic.

Distributed applications (dApps) written in languages like Go, Java, or Node.Js [23]. Specifically, the nodes can be: (i) Clients proposing transactions and broadcasting them to peers for ordering; (ii) Peers maintaining the ledger and the state of the latter; or (iii) Ordering service nodes that establish the order of transactions. Neither the execution nor the validation processes are performed by the latter. For implementing the application logic, Fabric uses smart contracts, known as chaincodes. A downside of this mechanism is that if one-third or more than one-third of the validators are not online, the system may halt. The Ethereum blockchain outperforms other blockchain networks as shown in Table 2.

TABLE 2. Distributed ledger technology (DLTs) comparison.

Features	Ethereum	Hyperledger	BitCoin
Platform nature	Generic blockchain platform	Modular blockchain platform	Financial industry
Means of operation	Permission-less, public	Permissioned, private	Permissioned
Support	Ethereum Developers	Linux	Bitcoin developers
Private Transaction Mode	No	Yes	No
Block Time	10-15 Sec	Depends on the peers node	10 Min
Transaction Limit	20 Trx/Sec	No	7 Trx/Sec
dApps	Yes	No	No
Censorship resistance	Yes	No	No
Scalability	No	Yes	No
Currency	ETH	None	BTC
Stimulus	Economics incentive	Reputational Risk	No
Scalability	No	Yes	No
Consensus	PoS	PBFT	PoW
Data Access	All Nodes	Authorized Nodes	Only relevant nodes
Cross-contracts	Yes	No	No

C. BLOCKCHAIN-BASED DATA STORAGE

With the advent of blockchain technology, decentralized storage systems (such as IPFS [24], Storj [25], Sia [26]) are used as blockchain-friendly off-chain mechanisms. Such systems store files in a distributed way without relying on centralized service providers. They provide free space for storage and rely on blockchain technology as their foundation.

In [27], the authors proposed a secure data-sharing framework for sensitive data. They used cryptographic techniques to access data. The sensitive record is uploaded to the blockchain network using an asymmetric encryption algorithm. A smart contract allows the user to have access to the data. However, there is no relevant regulatory mechanism in the proposed study. Moreover, the owner can no longer control the uploaded data once it is exposed to the viewer.

Reference [28] presented a data-sharing system between buyer and seller that enables privacy and open auditing techniques by employing IPFS, Ethereum, and encryption schemes to accomplish data security using fundamental aspects of blockchain such as decentralization, durability, and audibility. The proposed study catered to the storage problems by introducing IPFS, storing user data, and returning hash files. To achieve data security, the owner encrypts the hash file to overcome the risk of data threats. However, no key exchange mechanism is defined if the data owner loses his private key; PKG (private key generator) can still decrypt the server's data and perform data tempering. Moreover, the authors used RSA as an encryption scheme, which is computationally too costly.

Blockchain is also used for sharing medical records. Further research was conducted into effectively managing and protecting medical records. A blockchain-based data sharing strategy for patients' medical records is suggested, along with a decentralized record management system to handle EMRs [29]. The system provides digital protection for sharing data in cloud repositories. Asymmetric cryptography is adapted to encrypt the data. However, the proposed scheme does not take the concerning risk of sensitive data disclosure to the attackers. Moreover, the scheme does not propose a practical approach to address these challenges.

D. BLOCKCHAIN IN IOT

A blockchain-based architecture for personal data protection in the IoT has been developed, in which data is uploaded along with an attribute-based encryption system (ABE) [30]. For an efficient, lightweight, integrated blockchain for IoT devices to ensure data storage and privacy protection, the authors used a certificate-less cryptographic technique that reduced processing time and communication overhead. The proposed model achieved great success in IoT devices. However, public-key encryption (PKE) is computationally expensive for resource-constrained devices.

In [31] proposed an IFPS-based data storage mechanism in the IoT, the data along with access control policies were uploaded to the system. The system stored the encrypted file in chunks on each IPFS node. Shamir's secret sharing algorithm was used. Only authorized users have access to the stored data. The data file is dynamically linked with a consensus protocol. The confidentiality of the data in this study was excellently protected. However, this method can be applied to small data files and is unsuitable for large data sets.

To overcome the challenges mentioned above in blockchain-based data storage, [32] presented an IPFS-based framework. The proposed scheme ensures digital content preservation and storage issues. The encrypted data is uploaded to IPFS nodes, which generate a secret key for the data owner and return a hash file. An asymmetric key encryption algorithm is used for the encryption of data files. The registered user can request data, and the private key is shared among the data owner and user. Using the secret key, they can download the required data from IPFS. Due to the lack of a defined access policy, a malicious node can still access and control the data. Furthermore, the consensus mechanism is not exploited.

Use the paradigm of a multi-domain wireless sensor network (WSN) and game theory to examine the influence of

cooperative behavior in [6]. In the presented study, the participants are the various sensor nodes. The nodes are assumed to decide whether to help other nodes in data transmission or request other nodes to help transmission. The IPFS gateway is connected to the blockchain through smart contracts to provide file sharing access. The proof-of-work consensus mechanism is replaced with a less computationally expensive proof-of-possession mechanism. Although the proposed model has been thoroughly studied, the issue of the cooperative behavior of sensor nodes remains. In addition, no simulation results were provided in the proposed scheme to evaluate system performance or verify the claim.

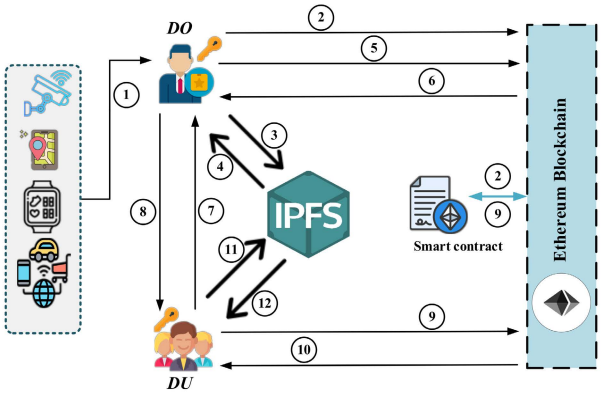

FIGURE 3. Our proposed IoTChain system model.

III. SYSTEM MODEL

Our presented IoTChain model consists of four entities: 1) IoT devices; 2) *DO*; 3) *DU*, and 4) Ethereum Blockchain and IPFS (storage module). The IoTChain system model is shown in Fig. 3.

1) *IoT Devices*, like environmental sensors, connected appliances, etc., may normally seek access and handle commands remotely. These devices are also in charge of data collecting, preliminary processing, and transmission. As these devices are resource-restricted, therefore, they send the collected data to storage providers.
2) *The Data Owner (DO)* is the individual or organization that owns the IoT data. They manage user queries and access by screening their requests.
3) *Data User (DU)* is the *DO* client that requests to view or download the data stored on IPFS nodes.
4) *Storage Module:* Ethereum Blockchain and InterPlanetary File System (IPFS) play a vital role in our system model, responsible for maintaining the whole system. The Ethereum blockchain ensures efficient and secure data sharing and storage using cryptographic techniques. IPFS is a decentralized storage system not directly related to Ethereum but can be integrated. IPFS works on a distributed hash table (DHT) for accessing files in the IPFS network. DHT locates the file through a content-based address. When uploading a file to IPFS, it will generate a unique cryptographic hash string like a fingerprint called content identifier (CID). This unique identification is known as URL on the web. While downloading this file, the computer asks IPFS if someone has the file with the particular cryptographic hash string and downloads the file from another node in the network. CID ensures that the right and non-tampered file has been sent to the user. CID is also helpful in avoiding multiple copies stored on IPFS, which turns the network more efficient and faster.

In our proposed IoTChain system model, *DO* first set up the system with master key DO_{MK}, deploy the smart contracts, and exchange the secret key as shown in Fig. 3. IPFS nodes are only accessible through a smart contract, and the double arrow pointing from IPFS and Ethereum blockchain shows their deployment.

① IoT devices collect the data from various monitoring areas. They send the processed data to the *DO*.
② *DO* uses master key DO_{MK} to initialize the process and embed DO_{MK} to Ethereum blockchain through a smart contract.
③ *DO* uses an AES-128 encryption scheme to encrypt the IoT data into CT_F, and then upload CT_F to the IPFS network.
④ IPFS returns the hash of the file (content identifier) to *DO* and records the file location $H_{Location}$ on IPFS network.
⑤ *DO* broadcast content identifier (CID) to the Ethereum blockchain for subsequent accessing and downloading of the data.
⑥ Ethereum blockchain returns transaction id (TX_{ID}), ABI code, and CID location CT_l to *DO*.
⑦ Once the data is uploaded and *DO* records all the information, *DU* sends a registration request to *DO*.
⑧ The system will authenticate *DU* through the user registration & Authentication portal and if the user is authentic. Then DO generates secret key DU_{SK}, and returns transaction id TX_{ID}, and file location $H_{Location}$ to *DU*.
⑨ *DU* searches for a smart contract and invokes it, read the transaction data, and relevant information on Ethereum blockchain i.e., transaction id (TX_{ID}), and CID location CT_l.
⑩ Ethereum blockchain returns transaction id (TX_{ID}), and CT_l based on smart contract search.
⑪ *DU* requests for the encrypted data (CT_F) from IPFS by providing the CID.
⑫ Finally, the encrypted file is downloaded and decrypted by *DU* secret key SK_{DU}.

A. PROTOCOL DETAILS

G_1 and G_2 are defined as two cyclic multiplicative groups of large prime integer ρ, and g $\in \mathbb{R}$. Let $\mathbb{Q}$ be the source of G_1 and G_2. G_T is the cyclic multiplicative group of the same order class, represented by 1; g is the element of $\mathbb{Q}$ that maps to G_T. The bilinear mapping $\hat{e}$: $G_1 \times G_2 \longrightarrow G_T$. Thus,

$\hat{e}$ is said to be a bilinear mapping if it meets the following criteria [33].

1) **Bilinearity:** Considering the variables a, b $\in \mathbb{Z}^* \mathbb{Q}$, and $\forall$ X $\in G_1$, $\forall$ Y $\in G_2$. As a result, $\hat{e}$ (a. X, b. Y) = $\hat{e}$ $(X, Y)^{a.b}$
2) **Nondegenerate:** Given two elements, X and Y, there is at least one element X such that $\hat{e}$ (X × X) $\neq 1$
3) **Computable:** Given two elements, $\forall$ X $\in G_1$, $\forall$ Y $\in G_2$, there is at least one efficient way to compute $\hat{e}$ (X, Y) G_2

In our proposed IoTChain model, the cost of bilinear mapping operation is high. Therefore, we choose fewer computation search function. The bilinear mapping $\hat{e}$ for our defined G_1 and G_2 cyclic multiplicative groups of large prime number ρ, will be $\hat{e}$: $G_1 \times G_2 \longrightarrow G_T$. We supposed that the user attribute set $\mathcal{S} = \{att_1, att_2 \ldots att_n\}$ has n attributes. Each attribute has multiple values, such as $\mathcal{S}_i = \{x_1, x_2 \ldots x_n\}$, therefore $\mathcal{S}_i$ and $\mathcal{S} = n_i$.

The two collision-resistant hash functions, $\mathbb{H}_1$: $\mathbb{Z}^* \mathbb{Q} \times \{0,1\}^{log_2^n} \times \{0,1\}^{log_2^m} \rightarrow \mathbb{Z}^* \mathbb{Q}$ and $\mathbb{H}_2$: $\mathbb{Z}^* \mathbb{Q} \rightarrow G_1$. The pseudo-random function f will be: $\{0,1\}^* \times \{0,1\}^{\lambda} \rightarrow \{0,1\}^l$. Also, DO select x and $y \in \mathbb{R}\ \mathbb{Z}^* \mathbb{Q}$, and computes $X_{i,k} = g^{\mathbb{H}_1(x||i||k)}$, and $Y_{i,k} = \hat{e}\ (g, g)^{\mathbb{H}_2(y||i||k)}$ for $1 \leq i \leq n$, and $1 \leq k_i \leq n_i$. Finally, the public parameter (PK) is public, and DO publish it on the media such as public database as $PK =< \rho, g, \{X_{i,k}, Y_{i,k}\}$ where, $< 1 \leq i \leq n, 1 \leq k \leq n_i >$. The encrypted master key $MK =< x, y >$ is embedded into the Ethereum transaction (TX_{MK}) and then DO deploy a smart contract on the Ethereum blockchain. Once the smart contract has been successfully deployed, record the ABI code and smart contract account address.

1) ATTRIBUTE BASED ACCESS CONTROL POLICY

A-BAC [34] implements an access policy $\mathcal{W}$, whose output is either 1 or 0, depending on the attribute set $\mathcal{S}$. According to A-BAC, $\mathcal{S}$ satisfies $\mathcal{W}$ if and only if $\mathcal{W}$ returns 1 [35]. Generally, the notation $\mathcal{S} \models \mathcal{W}$ denoted that $\mathcal{S}$ satisfies $\mathcal{W}$. In contrast, where $\mathcal{S}$ does not satisfy $\mathcal{W}$ represented by $\mathcal{S} \not\models \mathcal{W}$. In our proposed IoTChain model, AND-gate policy AND^*m are being considered. Formally, the access policy $\mathcal{W} = \{w_1, w_2 \ldots w_n\} = \lambda_i \in I_w$, where attributes list $\mathcal{S} = \{att_1, att_2 \ldots att_n\}$, and $I_w = \{i\ /\ 1 \leq i \leq n, \mathcal{W}_i \neq *\}$, we say $\mathcal{S} \models \mathcal{W}$ if $\mathcal{S} = \mathcal{W}_i$ for all $\{1 \leq i \leq n\}$ otherwise, $\mathcal{S} \neq \mathcal{W}$. It should be remembered that the wildcard $\mathcal{W}^*$ represents "don't care" values. For instance, if we have an access policy, $\mathcal{W} = \{Hospital.X, Physician*, \text{Pakistan}\}$, the attribute set $\mathcal{S} = \{Hospital.X, Physician, male, \text{Pakistan}\}$, and $\mathcal{S} = \{Hospital.X, Nurse, male, \text{Pakistan}\}$, then $\mathcal{S}_1 \models \mathcal{W}$, and $\mathcal{S}_2 \not\models \mathcal{W}$.

B. SYSTEM SETUP (1^{λ}) $\longrightarrow$ (PK, MK)

Our proposed solution combines A-BAC policy and AES-128 to ensure end-to-end data encryption and fine-grained permission in the distributed storage system. The result indicates that the data achieved fine-grained access control. In terms of cost and time, blockchain is an expensive medium for data storage. So, keeping the ciphertext on the Ethereum blockchain should be as short as possible to reduce the associated transaction cost. Smart contracts for data storage perform as few as possible calculations to reduce related computational costs, such as the A-BAC inverted index style approach implemented in [34]. The scheme is modified to support AND gate policy for user attribute set $\mathcal{S}$ and fixed ciphertext length.

DO starts the system configuration procedure by receiving the system security parameter λ as an input. It will return the system's public parameter PK and the master key K as outputs. Since PK is public and known to all users, DO places PK on public media such as websites, public databases, etc. At the same time, DO embeds K and deploys smart contracts on the Ethereum Blockchain. Further, the smart contract serves and stores encrypted keyword indexes and provides search services, as shown in Fig. 3 of the system model in steps ① and ②.

1) KEY GENERATION ($MK, PK, \mathcal{S}$) $\longrightarrow$ (SK_{DU})

The DO runs this algorithm. The process takes the master key MK, pubic parameter PK, and user attribute set $\mathcal{S}$ as an input to the system. We defined $\mathcal{S} = \{att_1, att_2 \ldots att_n\}$ as the $DU's$ attribute list that obtained the associated private key SK_{DU}. DO determines $SK_{DU} \in \mathbb{R}\ \mathbb{Z}^* \mathbb{Q}$ for each user. Then for $\{1 \leq i \leq n\}$, DO computes:

$$\overline{\sigma_i} = \sigma_{i,k} = g^{\mathbb{H}_1(y||i||k)} \times \mathbb{H}_2 SK_{DU}^{\mathbb{H}_1(x||i||k)}$$

Finally, it outputs the corresponding DU private key SK_{DU} with the associated attribute set $\mathcal{S}$, as shown in Fig. 3 of the system model in steps ⑦ and ⑧.

2) USER REGISTRATION ($K, \mathcal{S}$) $\longrightarrow$ (SK_{DO}, SK_{DU})

When a DU requests a file F, they need to be registered in the system before any other process. For registration purposes, they need to submit their own Ethereum account public key as input to the system, and the system will generate a unique ID for each user as an output. A smart contract will serve as a unique identifier for each user. A smart contract called "AllUsersMetadata" acts as a factory to produce a smart contract for each new user after they register. A public-private key pair is generated using the Elliptic Curve Digital Signature (ECDSA) algorithm provides registration key and current timestamp. The smart contract addresses are obtained from the registered user. The deployed user's smart contract contains the metadata, including the public key, registration key, and an array of information details regarding the files shared. When the user is authentic, DO assigns the attribute set $\mathcal{S}$ to each user. Thus, DO update the authorized user list by adding the user account address to "AllUserMetaData" in the smart contract as shown in Fig. 4. DO select secret key $SK_{DU} \in \mathbb{R}\ \mathbb{Z}^* \mathbb{Q}$, and assign to every user. Then $\{1 \leq i \leq n\}$, and attribute set $\mathcal{S} = \mathcal{S}_{i,k}$ to compute:

$$\overline{\Omega} = \Omega_{i,k} = g^{\mathbb{H}_1(x||i||k)} \times SK_{DU}^{\mathbb{H}_2(y||i||k)}$$

Finally, the respective attribute secret key is $SK_{DU} =< SK_{DU}\{\hat{\Omega}\}$ where $\{1 \leq i \leq n >\}$, search the secret key. $SK_{DU} =< K_s >$, $K_s \in \mathbb{R}$, and $\mathbb{Z}^* \mathbb{Q}$ where K_s is the same for every authorized user. *DO* share a secret key through elliptic curve Diffie-Hellman key exchange protocol [36]. In case of secret key distribution, *DO* embeds the encrypted keys into Ethereum transaction TX_{Encr} share his Ethereum account public key, transaction id (TX_{ID}), user attribute set $\mathcal{S}$, and smart contract source code to the user. During the IoTChain authentication process, the registration key and private key will be needed to verify the user's legitimacy. The detailed workflow of the user registration process is shown in Fig. 4.

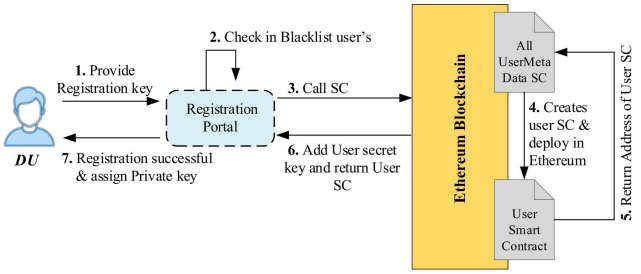

FIGURE 4. User registration process.

3) ENCRYPTION SCHEME

DO runs the encryption algorithm. In IoTChain model, we used AES-128 for encryption purposes. The prime benefit of AES-128 is that it provides fast and secure end-to-end encryption [37]. AES-128 is a symmetric key encryption algorithm with a key length size of 128 bits. It takes a data block of 128 bits as input that can be split into four operation layers. Each layer is represented as a 4×4 order of the matrix. The key size $(K) = \{128, 192, 256\}$ depends on the number of rounds $(N) = \{10, 12, 14\}$ for a full encryption and decryption process. The 128-bit replacement key is formed from the primary key in KAL (key addition layer), and it is XORed to each output of 1 byte to encrypt the data in a single cycle. Every round in AES uses substitution and permutation, which provides fast encryption and decryption operations and is appropriate for software and hardware level [38]. The encryption process is further divided into three sub-algorithms.

1) **File Encryption** $(F) \longrightarrow (CT_F, F_K, Kw)$
The *DO* runs the file encryption algorithm by taking the shared file F as input. The AES-128 encryption scheme will generate F into a ciphertext file (CT_F) as: $CT_F = Enc_K\ (F)$, where the original file F has been converted into an encrypted file and the file encryption key. The F is further used to locate the encrypted file location. *DO* upload the encrypted file CT_F to the IPFS and then return the file location $H_{Location}$ stored on a decentralized storage system that will later be used for file searching. As illustrate in Fig. 3 of system model of steps ③ and ④.

2) **Key Encryption** $(PK, F, H_{Location}) \longrightarrow (CT_{MD})$
The data owner runs the key encryption algorithm. Once the file F is uploaded and the ciphertext CT_F is generated, *DO* computes $CT_l = Enc_k\ (H_{Location})$, by taking public parameter PK, file encryption key F_K, and file location on IPFS node and access policy $\mathcal{P}$ as input. Then *DO* uses AES encryption to encrypt file encryption key K, under access policy $\mathcal{P}$ as:

$$< X_P, Y_P >=< \Pi_{i \in IP} \hat{X}_{i,k} \times \Pi_{i \in IP} \hat{Y}_{i,k} >$$

Where $< \hat{X}_{i,k}\ \hat{Y}_{i,k} >=< X_{i,k}\ Y_{i,k} >$, and IP is a subscript set of access policy $\mathcal{P}$. Then, *DO* randomly select s $\in \mathbb{R}$,and $\mathbb{Z}^* \mathbb{Q}$, and computes $CT_k =< \mathcal{P}. C_0, C_1, C_2 >$, where $C_0 = K. Y_s\ \mathcal{P}, C_1 = g_s, C\ \mathcal{P}_1 = X\ \mathcal{P}_s$. After the key encryption, *DO* randomly choose AES key K_1, and compute $CT_{MD} = Enc_K\ (CT_K, CT_{MD})$. *DO* embeds CT_{MD} into Ethereum transaction. Record transactions id TX_{ID} and related key K_1 once CT_F has been accepted. The key encryption procedure is depicted in Fig. 3 of steps ⑤ and ⑥.

3) **Keyword Search** *DU* runs the search algorithm by taking a keyword Kw and his secret key SK_{DU} as input to the system. The system searches for tokens as an output. *DU* read the relevant transaction data Kw from the Ethereum blockchain. Based on search token, *DU* selects keyword from the Ethereum blockchain to invoke a smart contract, as shown in Fig. 3 of steps ⑨ and ⑩.

C. DECRYPTION SCHEME

DU runs the decryption algorithm by using his own secret key SK_{DU}. It requires the file location on IPFS $H_{Location}$ ciphertext stream CT_l on the Ethereum blockchain, and the system public parameter PK and computes $\mathrm{d} = F(KW||1, K)$, then $TXID_j = d \oplus TXID_j$, and $K1_j = d \oplus K1_j$ for $TXID_j \in S_{TXID_j}$, and $K1_j \in S_{K1}$.

DU invokes a smart contract and reads relevant transaction $TXID_j$ data from the Ethereum network by using the AES algorithm to compute $(CT_l, CT_K) = \mathrm{DEC}K1_j\ CT_{MD}$. If the attribute set $\mathcal{S} = \mathcal{P}$, else, returns $\perp$ and reads the very next transaction details from the Ethereum network. *DU* locates $H_{Location}$ the $Enc_K\ (F)$ on IPFS and decrypts as:

$$DU_{(SK)} = \frac{C_0}{\hat{e}(\sigma_\rho, C_1) \times \hat{e}(\mathbb{H}_{2SK}, C_2)}$$

where $\hat{\sigma}_i = \sigma_{i,k} = g^{\mathbb{H}_1(y||i||k)}\ \mathbb{H}_2(SK)^{\mathbb{H}_1(x||i||k)}$, and computes $F = DEC_{CTF}$ to recover the original file F as shown in Fig. 3 of the system model in steps ⑪ and ⑫.

D. SMART CONTRACT DESIGN

This section mainly presents solidity smart contract-related interface and algorithm logic. The global namespace contains all of the unique variables and functions that are primarily used to provide information about the Ethereum blockchain. *msg.sender*: Transaction creator call. In an

Ethereum contract, this type of variable refers to the contract creator's address.

msg.value: It shows the amount of Wei sent with the transaction. A transaction's cost is represented by *msg.wei*, 1 ETH = 10^{18} Wei.

tx.origin: Transaction's initiator (full call chain). When a *DU* call multiple smart contracts, a E.O.A chain is generated, indicating that the E.O.A is *tx.origin*.

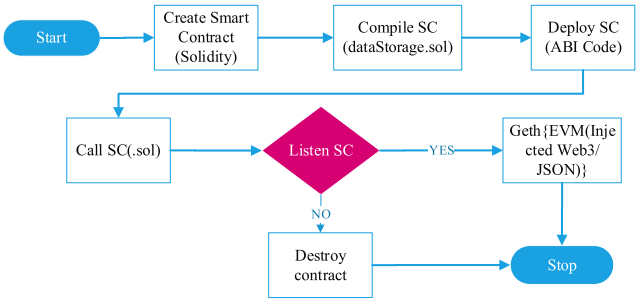

FIGURE 5. IPFS smart contract workflow.

1) IPFS SMART CONTRACT

DO deploy IPFS smart contracts, which we name *dataStorage* and *dataSharing*. The smart contract initialization process involves when this procedure defines various contract variables.

1) The address of the *DO* is defined as "dataOwner".
2) The mapping type "authorizedUser" variable defines a mapping collection from an authorized user address as a boolean value.
3) A mapping type specifies an index of encrypted keyword indexes to related data via a mapping variable. Smart contracts allow the *DO* to add, amend, and remove data collections. The authorized *DU* can access the data via smart contract interfaces.

Data storage and sharing: Ethereum smart contracts only provide log events to determine the return value of non-constant functions. Initially, the *DO* generates the original file F metadata to start the digital data sharing process. Metadata would include file name, type, size, and description. Consequently, in the above data sharing contract, the search outcomes returned by the search function are only accessible through events. In addition to the metadata, a complete encrypted file CT_F is uploaded to the IPFS. Here is Algorithm 1 of how files are uploaded to IPFS.

AddUser (new user account address): The contract's creator (*DO*) runs this algorithm by taking the user's identity (registration details) as an input to the function. The system authenticates the user through the registration portal as given in Fig. 4, and the system generates a private key for each user. The add new user algorithm is shown in Algorithm 2.

Update User: DO run this algorithm by taking user account address as input. *DO* update the user from the authorized set bypassing the user's EOA to the function. Here Algorithm 3 indicates the updated user account algorithm.

Algorithm 1: Incentive Based IPFS Data Storage & Sharing

Input: F

{ **Step 1:** *Upload IoT data to IPFS*}
  $Bytes_{32}[DataList]$
  Maps ($Bytes_{32} \implies$ Data) *DataMap*
  Maps ($Bytes_{32} \implies$ String) *ipfs.io*
  event $confirmFileIndex_{ID}()$
Function `SendDataToIPFS(`F`)`**:**
  Set DO ← *msg.sender*;
  Require ← *DUmetaMaskAdd*
  EthereumAccAdd = *await*
  web3.eth.getaccounts()
  SaveToIPFS awaitipfs.add ←
  (this.state.buffer,ipfshash)
  GenerateHash this.setstate ←
  ({*ipfshash.ipfshash*[0]})
  StoreHash AgainstId ←
  Storehash.methods.sendhash(*this.state.ipfs*)
  Transcation ←
  this.setState({*transcationhash*})
End Function
{**Step 2:** *Reward with Digital currency*}
  Coins ← *DOAssetsAvailable*
if *DOAssetsAvailable* < *tx.amount* **then**
  returnsErrormessage
  Else
  for *i* ← *0 to tx.amount -1* **do**
    Coins[*i*].*DO*← *asset.DO*
    Update AssetRegistery
  end
end
{**Step 3:** *Event generation*}
  Emit event of sharing
  Update asset status
  Return Sharing successful

RemoveUser (old account address): DO run this algorithm. A malicious user needs to be removed from the authorized user's list by providing his/her E.O.A to the function as shown in Algorithm 4.

Delete File (remove unwanted file): Only the contract's creator (*DO*) can execute this function by taking the encrypted keyword index of the file (*keywordIndex*) and its associated transaction id (TX_{ID}) as input to the function as shown in Algorithm 5.

IV. RESULT AND DISCUSSION

A. SIMULATION ENVIRONMENT

In this section, we implemented a prototype to analyze the desired performance of our proposed IoTChain model. The specific system configuration is an Intel Core i5 @ 3.6Hz Processor, 8GB of RAM, and a 64-bit operating system to execute experimental tasks. We used the Ethereum blockchain

Algorithm 2: Add New User to IPFS

INPUT: $Bytes_{32}ID$, *Credential*
OUTPUT: *BOOLEAN*
if *msg.sender is NOT DO* **then**
  throw;
  if *msg.sender is NewUserAddr* **then**
    RegisteredUserList [*NewUserAddr*]
    return *true*;
  else
    [*NewUserAddr*] exist;
    return *false*;
  end
end

Algorithm 3: Update Existing User

INPUT: *AuthorizedUserList*
OUTPUT: *BOOLEAN*
if *msg.sender is NOT DO* **then**
  throw;
  if *map authorizedUserslist*[*i*] ← *false* **then**
    return *false*;
  else
    map authorizedUserslist[*UpdateUser*] ← *true*;
    return *true*;
  end
end

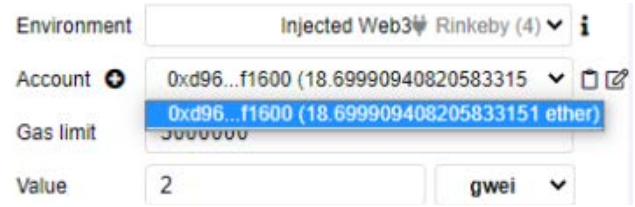

FIGURE 6. Smart contracts account address.

to perform simulations. In terms of the number of transactions verified per second, Ethereum outperforms the Bitcoin blockchain. Solidity [39], a Turing-complete scripting language, is used for writing smart contracts. Remix [40] is an online IDE and is used to execute smart contracts. Ganache is a personal blockchain network with virtual accounts with unique account addresses that provides developers with 100 test Ether when linked to Metamask. MetaMask [41], an online wallet (i.e. a place to store cryptocurrency) used by Truffle to run contracts. The Ethereum wallet delivers virtual Ether for testing. In the development environment, Metamask pays the computational costs. In addition to the Truffle developer environment, Solidity is used as the primary programming language to create smart contracts. Smart contracts can add, update, maintain, and modify digital transactions. Additionally, we use web3.js to generate and deploy the proposed smart contracts. For the EVM to

Algorithm 4: Remove Old Account

INPUT: *UserAccountAddress*
OUTPUT: *BOOLEAN*
if *msg.sender is NOT DO* **then**
  throw;
  if *OldUserAddr does not exist* **then**
    return *false*;
  else
    registeredUsers[*OldUserAddr*] ← *false*;
    return *false*;
  end
end

Algorithm 5: Delete File From IPFS

INPUT: *KeywordString*, TX_{ID}
OUTPUT: *NULL*
if *msg.sender is Not DO* **then**
  throw;
end
getValueIndex[*keywordString*] *length*;
for *length*[*i*] *to length* − 1 **do**
  if *Index[stringIndex][i].*TX_{ID} = TX_{ID} **then**
    for *K* ← *i* + 1 *to length -1* **do**
      Push ← Index[*stringIndex*] [*K* − 1];
      Index[*stringIndex*][*K*]
    end
    delete index [*stringIndex*] [*length* − 1];
    break;
  end
end

TABLE 3. IPFS smart contract cost test, 1 Gas unit = 2 Gwei, 1 ETH = $4,290.

IPFS Functions	Transaction Cost(Gwei)	Execution Cost(Gwei)	Actual Cost(ETH)
IPFS dataStorage contract create	328144	297656	0.000831
IPFS dataSharing contract create	304076	214378	0.000544
deleteFile	39743	27645	0.000116
addUser	63897	41025	0.000113
updateUser	40205	24797	0.000035
removeUser	26656	14308	0.000024

work properly, these components must work collectively. A blockchain is dependent on scripts to control data flow between *DO* and *DU*. All the smart contracts are deployed

TABLE 4. System security cost test (GasPrice = 2 Gwei, 1 ETH = $4,290).

Security Parameters	Size (bytes)	Gas Used	Actual Cost (ETH)
Master key (MK)	153	19305	0.000043921
DU secret key (SK_{DU})	908	71554	0.000097358
Cipher-Text (CT_F)	538	48319	0.000081568
Proof-of-Work (PoW)	—	924893	0.002314
Proof-of-Authority (PoA)	—	589483	0.001736

on the Rinkeby test network, the official Ethereum test network [42].

For cryptographic purposes, we use the hash functions $\mathbb{H}_1$ and $\mathbb{H}_2$, which are from the Miracle library [43], and the curve is the Cocks-Pinch curve. *DO* runs the AES-128 encryption algorithm by taking file F as input into the cipher-text stream CT_F. The user attributes are set to 4.

To run the corresponding smart contracts, we set the gas price to be adjusted to 2 Gwei. Wherein 1 Gwei is equivalent to 10^9 ETH. The given formula measures the actual transaction fee:

$$ETH = GasUsed \times GasPrice$$

The gas consumption and corresponding cost measured for smart contracts are given in Table 3. A smart contract for the review system is deployed on the Ethereum platform. Gas is required to deploy any contract. Gas is limited by a maximum predetermined by the creator, which is 3,000,000. Whenever a contract is deployed for the first time, the level of intensity is higher, which makes the limit higher based on the block size and the miner's fee. Gas consumption is used to calculate the execution and transaction costs. There are two kinds of gas consumed: transaction gas and execution gas. The *transaction cost* is the amount of gas needed to perform any action on the blockchain network, while the *execution cost* is the computing price required to execute the smart contract. To successfully execute smart contracts on the blockchain, we initially had to specify the computational cost limit for the initiation and completion of transactions. A pre-defined quantity of gas consumption is charged for each transaction as given in the Ethereum yellow paper [10]. In our experiments, gasPrice was set to 2Gwei, where 1Gwei $= 10^9$ wei $= 10^{-9}$ ETH.

$$1 \text{ Gas Unit} = 2 \text{ Gwei}(1 \text{ ETH} = 10^9 \text{ Gwei})$$

The gas consumption and $cost for the various IPFS smart contracts and functions are listed in Table 3. The transaction and execution cost for the IPFS *dataStorage* smart contract were recorded at 328144 and 297656 respectively, which were noticed to be almost unchanged upon multiple executions, and the associated $cost were $3.56. The *dataSharing* contract was created only once, and the $cost was $2.33 as shown in Fig. 7. When a *DU* requests the data, the *addUser* operation needs to be performed. Similarly, when a *DO* removes a specific user from the authorized user's list by calling *removeUser* function, the user's account address is stored in the blocklist for future reference. The two functions' costs (USD) were $0.48 and $0.10, respectively as shown in Fig. 8.

When *DO* deletes a file from the system, *deleteFile* operation is invoked. The $cost associated with this operation was $0.49. We set the cost to 0.01 ETH. The *DO* can regularly verify the balance of the data-sharing contract. When the credit exceeds zero, a withdrawal procedure can be used to transfer it to the creators of an externally owned account (E.O.A). Additionally, the $costs for transaction fees may change depending on the number of files; therefore, we set the fixed number of files. The IPFS contract functions were tested, and the results were recorded. The $cost for *dataStorage* operation is high as more data is added to the system. When *DO* performs *deleteFile* operation, the $cost is low for the first file deletion. When the number of files increases, the transaction fees also increase accordingly. These IPFS smart contracts were deployed on injected web3 environment, and the test network was the Rinkeby with the given account address. The *DO* account address is 0xd96dfe18b6daf5ec36d15a0e7 a61811afd4f1600 and *DU* account address is 0×081dc135b8cef8b6efc5a9134 4de7f10b373e1b2. These simulated results are online and can be seen at account address 0xd96dfe18b6daf5ec36d15a0e 7a61811afd4f1600 on https://rinkeby.etherscan.io/ as shown in Fig. 6.

We implemented a BASE 64 encoding scheme and converted each data chunk to JSON style to make the results more understandable. Some of the costs of the smart contract measured by the experiment are shown in Table 4. Furthermore, the *DO* sets up the system with a master key (MK), which is constant and does not change for every user. Moreover, MK should be used only once to invoke the process. The master key size for the experiment is 153 bytes, with an associated $cost of $0.543. While the *DU* secret key size is the largest, which is 908 bytes, and depends on the number of user's attributes, the $cost was $0.045. Fortunately, the system only needs the *DU* secret key once for each user. The number of user attributes is 4, which is fixed. The ciphertext length (CT_F) is also constant. The measured value is 538 bytes, and $cost was $0.0359. The CT_F must be saved only once for each shared file.

The user's trust is an essential factor in sharing environment. Our solution is practical and suitable for resource-constrained IoT devices that require storage and

energy. The *dataStorage* and *dataSharing* smart contracts were implemented on EVM. Their transaction and execution costs are recorded as low as expected, showing that our solution is feasible for lightweight IoT devices. Transaction cost is the amount of gas consumed for deploying the contract on EVM. Where execution cost depends on the logical operation being performed and the number of lines of code and is always lower than transaction cost.

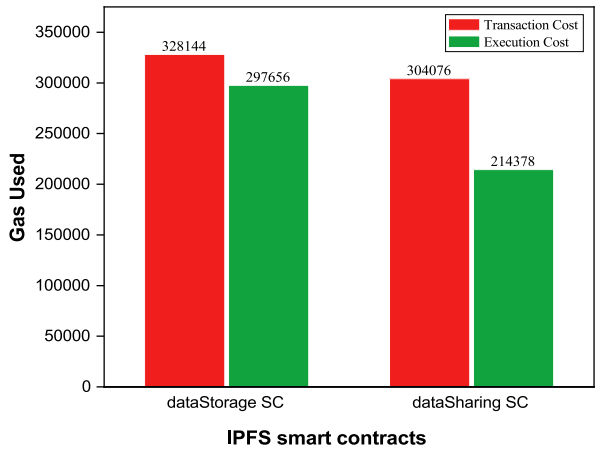

FIGURE 7. IPFS *data storage* & *data sharing* smart contracts.

In Fig. 7 the *dataStorage* smart contract consumed more gas than *dataSharing* shows that data upload to IPFS, the Ethereum blockchain performs more operations. This is an intensive process that incurs high gas consumption. We found that the higher the limit, the more gas is used. As a result, intensity is a crucial factor in this stage. In these smart contracts, we send IoT data through a function called "*SendDataToIPFS*," and an IPFS gateway known as "*ipfs.io*" that sends IoT collected data, such as pictures, videos, files, etc., to IPFS in an encrypted format. As a result, IPFS will send a hash file known as a content identifier (CID) that we stored on the Ethereum blockchain network using a particular function, "*StoreHashonEth*". The system is only accessible to an authentic user, which invokes smart contracts and gets the hash of the data, using the CID and downloading the data from the IPFS server. IPFS works like standard internet.

Fig. 8 represents gas used for various IPFS functions. The function *addUser* adds new users to the registered user list who request IoT stored data. Their transaction and execution costs were 63897 and 41025, respectively, and gas consumption was higher than other functions. While the *deleteFile* function locates the file, after the *DO* confirmation, this function deletes the file from the IPFS server, and the associated hash is deleted from the Ethereum blockchain. The *removeUser* function required less gas as compared to the other functions, which were recorded at 26656 and 14308, respectively. The implementation of computational costs and limitations in smart contracts is another challenge based on how much gas is used. Gas limits are the maximum computational costs that we are willing to incur for this experiment by spending money on transactions. Therefore, the challenge in this context is setting a reasonable limit to execute the transaction without failure. The more complicated the transaction, the more computer labor is necessary, as demonstrated by the experiment.

In the experimental result, we deployed the smart contracts on the Ropsten test network and Rinkeby test network, respectively. The Ropsten test network is used for a proof-of-work (PoW) consensus mechanism, and the Rinkeby test is used for the proof-of-authority (PoA) consensus mechanism. After several simulations, we concluded that our proposed PoA consensus mechanism required lower gas consumption and less execution time to add the newly generated block to the blockchain. Therefore, we clinched that our projected PoA consensus mechanism is superlative in IoT data storage and transmission, as shown in Fig. 9. The recorded gas consumption for PoA and PoW was 589483 and 924893, respectively. Furthermore, the transaction execution time for PoA and PoW was recorded at 487.6 (ms) and 678.4 (ms).

In addition, PoW is energy-intensive, adds to environmental stress, generates negative media attention, and has a high transaction fee, making it unviable for a long period. We offer a PoA model that is capable of processing more transactions per second. As a result, PoA networks are safe since nodes are chosen at random. Our experimental results concluded that PoA is the ideal model for our suggested scenario since it is a highly secure and energy-efficient consensus mechanism. Therefore, we chose the PoA consensus mechanism, which required less execution time and lower gas consumption.

In IoTChain model, we used the AES encryption algorithm, which has the advantage of being simple, parallel processing, error-proof, and impossible to decrypt. The comparison used different file sizes and different key lengths as shown in Fig. 10. The AES 128 algorithm offers 2^{128} keys, while the AES 256 algorithm offers 2^{256} keys. The larger the combination of keys, the longer the computation time. The execution time and threat attacks were used to test the effectiveness of each cryptographic technique. Each experiment employed five cryptographic algorithms (AES-128, DES-256, 3DES-168, RC2-128, and RSA-2048) using six text files (910kB, 5.4MB, 11.8MB, 35.6MB, 59.8MB, 106MB, and 256MB). Each algorithm's performance evaluation was performed based on its speed, memory file size, and throughput statistics using the advanced encryption package 2019. The encryption throughput is determined by dividing the calculated encrypted plain-text (in bytes) by the calculated encryption time (in ms). AES records one of the least encryption times, whereas RSA requires more computational time, respectively. These two schemes have significant differences in computational time because of the size of their search spaces. Fig. 10 shows AES-128 supremacy over other encryption algorithms in execution time and better security. Since DES is less time-consuming than other algorithms, it is the preferred algorithm after AES. 3DES and RC2 usually take the same time to perform the

encryption process; however, RSA is the slowest. Thus, it was evident that AES-128 is the fastest algorithm for encryption and decryption. It can be seen from the test results that the AES encryption algorithm is suitable for IoT data security.

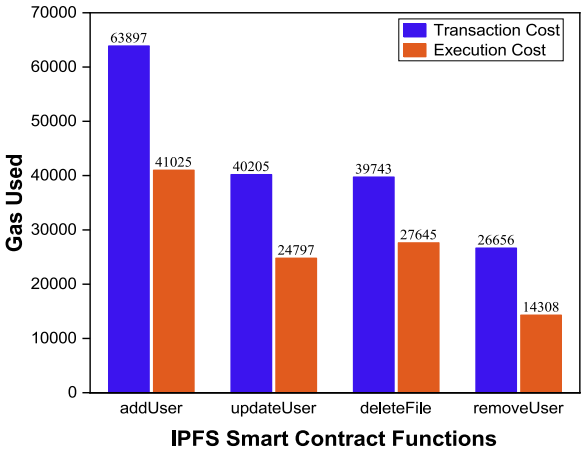

FIGURE 8. IPFS smart contract functions.

V. CHARACTERIZATION OF IOTCHAIN SCHEME

Our proposed IoTChain model combined Ethereum blockchain with IPFS, a distributed and reliable storage system, smart contract technology, and the most secure and fastest encryption technique, AES-128, to protect IoT data. IoT data storage is addressed while ensuring confidentiality, non-tampering, and gaining advantages over cloud storage systems. Data from IoT devices is stored in IPFS, and the returned hash code is encrypted before being published as a stream cipher in the blockchain. To get the IPFS hash code, registered users initiate the smart contract on the Ethereum blockchain. For security testing, we used the Ethereum-friendly software Oyente. Oyente produces a report of the most likely security risks. IoTChain is safe from possible threats, such as Integer Overflow, Parity Multisig Bug 2, and Call Stack Depth Attack. Our solution is tested against the attacker model.

A. OFF-CHAIN DATABASE STORAGE AND IOT INTEGRATION

Data collected by the IoT, such as photographs, and movies, require much memory. Our solution ensures data availability and stability by redundancy backup method, error-coding, and a FileCoin incentive mechanism. IPFS is a peer-to-peer file system that combines the distributed hash table (DHT) routing mechanism with BitTorrent technology to achieve quicker data throughput and lower costs. This study offers an external chain database for storing IoT data synchronized with Ethereum and IPFS. The actual data captured by the IoT device is saved mainly in an external database known as IPFS, and the hashes created by the device are stored on the Ethereum blockchain. The smart contract is invoked by a legitimate user who obtains the hash. They will download the required files from IPFS using these hashes. For the experimental analysis, the *DO* uploaded the IoT data to IoTChain model, we received the hash value for the file returned by the system. AES encryption is used to encrypt the hash value, and the result is shown in Table 5. Performance testing and functional testing of smart contracts were conducted on the Rinkeby Testnet test network. An AES encrypted ciphertext is permanently stored on the blockchain by way of the smart contract. The detail of the contract deployment is given in Table 5.

B. ACCESS CONTROL SCHEME

We propose an Attribute-based Access Control (A-BAC) policy as a means to apply a blockchain-based ABE (attribute-based encryption) scheme to the IoT data using a consensus-driven approach. A-BAC is a new access control mechanism where the data owner decides which of the attributes in its domain should be assigned to a user list. For the encryption of ciphertext, the data owner may combine different attributes from multiple attribute lists in keys for decryption and access policies. A central key management system assigns users attributes and gives them individual private keys. The novelty of our protocol is the addition of a Blockchain, which makes the attribute to user mapping more private and secure from a single authority to a distributed ledger. Through the blockchain, all users and attributes in the system can be represented in one place, resulting in a reliable, traceable chain of delegated access rights. Everyone has access to the blockchain, which provides proof of decentralized trust. A salient advantage of the A-BAC policy is that it offers cryptographic solutions to problems solved by traditional access control systems. In this way, the data is publicly accessible, but legitimate users can only decrypt it.

Unlike previous approaches, we propose to mitigate key revocation and management issues through the use of a distributed infrastructure, a distributed Blockchain. This paper presents an efficient construction of the distributed A-BAC scheme and introduces how the core operations can be integrated into the Ethereum blockchain to manage keys and attributes efficiently.

C. COMPARISON OF SCHEMES CHARACTERISTIC

We also discussed some parameters included in our framework and used them to evaluate against related work. In Table 6, we present assessments between some existing schemes in terms of decentralization, distributed data storage and sharing, data encryption, keyword search, verifiable results, and access control as we achieved in our IoTChain model. It is also essential that these parameters be implemented within the framework while maintaining security and privacy. We used the symbol "✓" to refer to the scheme with this feature, whereas "✗" indicates an opposite condition. The Table 6 indicates that schemes [28] and [30] do not meet the feature of data control, data encryption and keyword search of the stored data. While the scheme [31] and [33] support off-chain data storage, data sharing, and access data

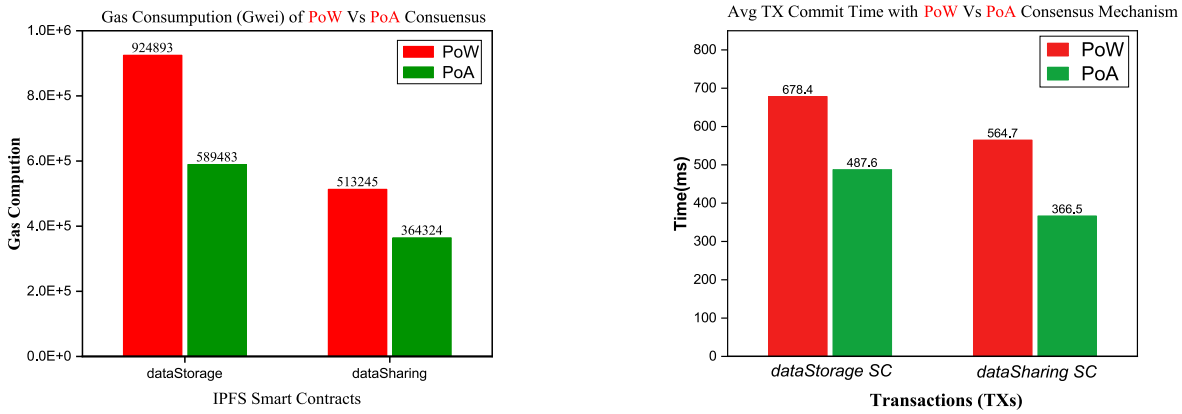

FIGURE 9. Comparison of PoW & PoA consensus mechanism in term of Gas consumption & time (ms) taken.

TABLE 5. Data upload to IPFS and ciphertext upload to ethereum blockchain network.

Key	Value
Status	0x1 *Transaction mined and execution succeed*
Block	9702871 (45790) *Block Confirmations*
Timestamp	7 days 22 hrs. ago (Nov-25-2021 07:46:01 AM +UTC)
From	0xd96dfe18b6daf5ec36d15a0e7a6181 1afd4f1600
To	[Contract 0xd81374b8a34f4e05 1f58c4545e5d6f9290 ff72e9 Created]
Transaction Hash	0x83f0ad39240b1a1a67b6b3d765d23 beeacef759ebda2e90162989c6defc2
Ethereum Contract address	0xd96dfe18b6daf5ec36d15a0e7a618 11afd4f1600
DataHash in IPFS	DataStorage.SendDataToIPFS(bytes32,string) 0xda660579af6f790f70231c7e099b2f48374da23
Input	0xbac3fef4a5b4682c86fe97973f765eb cc9fe41a1a8eede69e971df39360a0a92
Transaction Fee	0.001500867017357526 Ether
Gas Price	0.000000001825345998 Ether (1.825345998 Gwei)
Gas Limit & Usage by Tx_n	822,237 822,237 (100%)

TABLE 6. Comparison with some existing studies.

Reference	Blockchain	IPFS	IoT Data	Data Encryption	Data Sharing	Fine-Grained Permission	Data Control	Keyword Search
[28]	✓	✓	×	✓	✓	✓	×	×
[30]	✓	×	✓	×	✓	×	×	×
[31]	✓	✓	×	×	✓	×	✓	✓
[32]	✓	×	✓	✓	×	×	×	✓
[33]	✓	✓	✓	×	✓	×	✓	×
[34]	✓	×	✓	×	×	✓	✓	×
IoTChain	✓	✓	✓	✓	✓	✓	✓	✓

control; schemes [32] and [34] do not support data sharing and keyword search. Additionally, our scheme stores the data in IPFS, effectively solving data loss or tampering within the cloud environment. Moreover, only we have a scheme that meets all the properties and is more suitable for current times.

D. SINGLE POINT OF FAILURE

The proposed IoTChain model is compared to traditional data storage services; our solution overcame the single point of failure issue. It provides backup policy, reliability, and accessibility of IoT data, ensuring Proof-of-Replication and the IPFS incentive mechanism. In addition, IPFS is

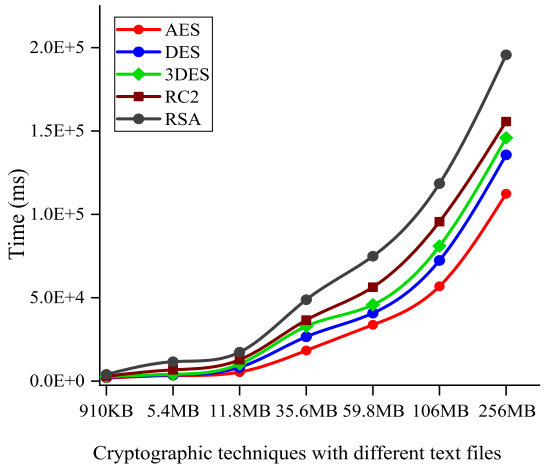

FIGURE 10. Time evaluation of different cryptography.

running peer-to-peer via the DHT routing, and the BitTorrent protocol lowers the costs more than the traditional cloud service.

E. SECURITY AND PRIVACY THREATS

Large files collected by IoT devices, including photographs and movies, are chunked and saved in IPFS on several storage nodes in our system. These files are encrypted using AES and stored in IPFS storage nodes. The storage nodes can only see a subset of the cipher-texts and have no access to any file metadata. It is difficult for an adversary to tamper with the IPFS data or create a single point of failure if they have access to it. The data on IPFS is secured using a sophisticated cryptographic technique (AES). Only the user's private key will be used to decode the data. The secret key is shared among *DO* and *DU* via the elliptic curve Diffie-Hellman key exchange protocol, making it impossible to determine the key for an attacker node. If the Ethereum blockchain and the ABE scheme are safe, the proposed scheme can also be considered secure.

F. DATA CONFIDENTIALITY

The IoT data is encrypted and saved on IPFS nodes. Only authorized users with their private keys have access to the encrypted content. The attacker nodes are eliminated by the use of an AES encryption mechanism. As a result, the data is protected, and only authorized individuals can access it. Through smart contracts, we proposed a scheme to ensure the fairness of the search process and the operations are performed honestly and by predefined logic.

VI. CONCLUSION AND FUTURE WORK

As IoT devices surge, data storage management, data accessibility, data transparency, and data privacy become essential considerations. Traditional storage methods may render data unavailable due to circumstances such as force majeure (political censorship, single point of failure, natural disaster). As a result, we propelled a ground-breaking blockchain-based IoT information paradigm. We named it the IoTChain Model. It allows for large-scale, safe storage of IoT information and accessibility to legitimate users. It also offers several advantages over a centralized system, such as low cost and high throughput. In this study, we explore the difficulties of IoT data storage and sharing. Our presented study combines distributed storage known as IPFS, the Ethereum blockchain, the AES encryption method, and a gas-efficient consensus mechanism. There is no need for a trustworthy PKG. We designed a blockchain-friendly off-chain mechanism to store actual IoT data. We developed fast and complex authentication, secret protection, and multi-signature-based conditional provenance approaches that allowed us to instantly access rights, manage, and limit data on the Ethereum blockchain. Experimental results demonstrated that our system provides durable, comprehensive, and tamper-resistant data management services. According to the simulation findings, adopting a proof-of-work (PoW), consensus mechanism instead of proof-of-authority (PoA) decreases 20-25% of gas usage. Furthermore, the AES-128 presented the fastest by 65% of all encryption strategies among different cryptographic approaches, yet the safest and secure. Our experimental analysis shows that the scheme is rational and feasible.

However, our approach does not support the functions of user attribute revocation and A-BAC policy updating. Furthermore, we will intensify our research efforts to make IoT data trading and administration easier using Ethereum native currency known as Ether (ETH). This is our next research goal.

REFERENCES

[1] M. A. Ferrag and L. Shu, "The performance evaluation of blockchain-based security and privacy systems for the Internet of Things: A tutorial," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17236–17260, Dec. 2021.

[2] M. Haghi Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, "A systematic review of IoT in healthcare: Applications, techniques, and trends," *J. Netw. Comput. Appl.*, vol. 192, Oct. 2021, Art. no. 103164.

[3] J. Guo, X. Ding, and W. Wu, "Reliable traffic monitoring mechanisms based on blockchain in vehicular networks," *IEEE Trans. Rel.*, early access, Jan. 13, 2021, doi: 10.1109/TR.2020.3046556.

[4] V. Hemamalini, G. Zayaraz, and V. Vijayalakshmi, "BSPC: Blockchain-aided secure process control for improving the efficiency of industrial Internet of Things," *J. Ambient Intell. Hum. Comput.*, pp. 1–14, Jan. 2022.

[5] A. Lopez-Vargas, A. Ledezma, J. Bott, and A. Sanchis, "IoT for global development to achieve the united nations sustainable development goals: The new scenario after the COVID-19 pandemic," *IEEE Access*, vol. 9, pp. 124711–124726, 2021.

[6] R. Gürfidan and M. Ersoy, "A new approach with blockchain based for safe communication in IoT ecosystem," *J. Data, Inf. Manage.*, pp. 1–8, Feb. 2022.

[7] P. I. R. Grammatikis, P. G. Sarigiannidis, and I. D. Moscholiosb, "Securing the Internet of Things: Challenges, threats and solutions," *Internet Things*, vol. 5, pp. 41–70, Mar. 2019.

[8] I. Makhdoom, I. Zhou, M. Abolhasan, J. Lipman, and W. Ni, "PrivyShare: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101653.

[9] P. Nerurkar, D. Patel, Y. Busnel, R. Ludinard, S. Kumari, and M. K. Khan, "Dissecting bitcoin blockchain: Empirical analysis of bitcoin network (2009–2020)," *J. Netw. Comput. Appl.*, vol. 177, Mar. 2021, Art. no. 102940.

[10] G. Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Byzantium Version. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[11] Y. Zhou, J. Wu, and S. Zhang, "Anonymity analysis of bitcoin, zcash and ethereum," in *Proc. IEEE 2nd Int. Conf. Big Data, Artif. Intell. Internet Things Eng. (ICBAIE)*, Mar. 2021, pp. 45–48.

[12] P. A. Lobo and V. Sarasvathi, "Distributed file storage model using IPFS and blockchain," in *Proc. 2nd Global Conf. Advancement Technol. (GCAT)*, Oct. 2021, pp. 1–6.

[13] J. Chi, Y. Li, J. Huang, J. Liu, Y. Jin, C. Chen, and T. Qiu, "A secure and efficient data sharing scheme based on blockchain in industrial Internet of Things," *J. Netw. Comput. Appl.*, vol. 167, Oct. 2020, Art. no. 102710.

[14] S. K. Dwivedi, R. Amin, and S. Vollala, "Blockchain-based secured IPFS-enable event storage technique with authentication protocol in VANET," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 12, pp. 1913–1922, Dec. 2021.

[15] P. Xiaoqiong, Y. Ting, C. Wenjun, W. Yunting, and L. Tianye, "Digital rights management scheme based on secret sharing in blockchain environment," *J. Comput. Appl.*, vol. 41, no. 11, p. 3257, 2021.

[16] S. Alam, M. Shuaib, W. Z. Khan, S. Garg, G. Kaddoum, M. S. Hossain, and Y. B. Zikria, "Blockchain-based initiatives: Current state and challenges," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108395.

[17] T. Wang, C. Zhao, Q. Yang, S. Zhang, and S. C. Liew, "Ethna: Analyzing the underlying peer-to-peer network of ethereum blockchain," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2131–2146, Jul. 2021.

[18] T. Frikha, A. Chaari, F. Chaabane, O. Cheikhrouhou, and A. Zaguia, "Healthcare and fitness data management using the IoT-based blockchain platform," *J. Healthcare Eng.*, vol. 2021, Jul. 2021, Art. no. 9978863.

[19] J. Bauvars, "Applicability of blockchain technology in securities settlement," *Complex Syst. Informat. Model. Quart.*, no. 28, pp. 34–58, Oct. 2021.

[20] A. H. Mohammed, A. A. Abdulateef, and I. A. Abdulateef, "Hyperledger, ethereum and blockchain technology: A short overview," in *Proc. 3rd Int. Congr. Human-Comput. Interact., Optim. Robotic Appl. (HORA)*, Jun. 2021, pp. 1–6.

[21] C. Chen, J. Yang, W.-J. Tsaur, W. Weng, C. Wu, and X. Wei, "Enterprise data sharing with privacy-preserved based on hyperledger fabric blockchain in IIOT's application," *Sensors*, vol. 22, no. 3, p. 1146, 2022.

[22] Y. Chen, M. Li, X. Zhu, K. Fang, Q. Ren, T. Guo, X. Chen, C. Li, Z. Zou, and Y. Deng, "An improved algorithm for practical byzantine fault tolerance to large-scale consortium chain," *Inf. Process. Manage.*, vol. 59, no. 2, Mar. 2022, Art. no. 102884.

[23] K. B. Jyothilakshmi, V. Robins, and A. S. Mahesh, "A comparative analysis between hyperledger fabric and ethereum in medical sector: A systematic review," in *Sustainable Communication Networks and Application*. Singapore: Springer, 2022, pp. 67–86.

[24] E. Daniel and F. Tschorsch, "IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 31–52, 1st Quart., 2022.

[25] S. de Figueiredo, A. Madhusudan, V. Reniers, S. Nikova, and B. Preneel, "Exploring the storj network: A security analysis," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, 2021, pp. 257–264.

[26] N. Zahed Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *J. Netw. Comput. Appl.*, vol. 162, Jul. 2020, Art. no. 102656.

[27] M. J. Hossain Faruk, H. Shahriar, M. Valero, S. Sneha, S. I. Ahamed, and M. Rahman, "Towards blockchain-based secure data management for remote patient monitoring," in *Proc. IEEE Int. Conf. Digit. Health (ICDH)*, Sep. 2021, pp. 299–308.

[28] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[29] A. Al Mamun, F. Jahangir, M. Umor, S. Azam, M. S. Kaiser, and A. Karim, "A combined framework of interplanetary file system and blockchain to securely manage electronic medical records," in *Proc. Int. Conf. Trends Comput. Cogn. Eng.* Singapore: Springer, 2021, pp. 501–511.

[30] J. Wang, J. Chen, N. Xiong, O. Alfarraj, A. Tolba, and Y. Ren, "S-BDS: An effective blockchain-based data storage scheme in zero-trust IoT," *ACM Trans. Internet Technol.*, 2022.

[31] M. Naz, F. A. Al-zahrani, R. Khalid, N. Javaid, A. M. Qamar, M. K. Afzal, and M. Shafiq, "A secure data sharing platform using blockchain and interplanetary file system," *Sustainability*, vol. 11, no. 24, p. 7054, 2019.

[32] R. Goyat, G. Kumar, R. Saha, M. Conti, M. K. Rai, R. Thomas, M. Alazab, and T. Hoon-Kim, "Blockchain-based data storage with privacy and authentication in Internet-of-Things," *IEEE Internet Things J.*, early access, Aug. 24, 2020, doi: 10.1109/JIOT.2020.3019074.

[33] T. Li, H. Wang, D. He, and J. Yu, "Blockchain-based privacy-preserving and rewarding private data sharing for IoT," *IEEE Internet Things J.*, early access, Jan. 31, 2022, doi: 10.1109/JIOT.2022.3147925.

[34] W. Xiang and Z. Yuanyuan, "Scalable access control scheme of Internet of Things based on blockchain," *Proc. Comput. Sci.*, vol. 198, pp. 448–453, Jan. 2022.

[35] X. Lu, S. Fu, C. Jiang, and P. Lio, "A fine-grained IoT data access control scheme combining attribute-based encryption and blockchain," *Secur. Commun. Netw.*, vol. 2021, Sep. 2021, Art. no. 5308206.

[36] S. Mitra, S. Das, and M. Kule, "Prevention of the man-in-the-middle attack on Diffie–Hellman key exchange algorithm: A review," in *Proc. Int. Conf. Frontiers Comput. Syst.* Singapore: Springer, 2021, pp. 625–635.

[37] M. Bedoui, H. Mestiri, B. Bouallegue, B. Hamdi, and M. Machhout, "An improvement of both security and reliability for AES implementations," *J. King Saud Univ.-Comput. Inf. Sci.*, Jan. 2022.

[38] A. Nurgaliyev and H. Wang, "Comparative study of symmetric cryptographic algorithms," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Oct. 2021, pp. 107–112.

[39] *Units and Globally Available Variables*. Accessed: Jan. 7, 2022. [Online]. Available: https://docs.soliditylang.org/en/latest/units-and-global-variables.html

[40] *Ethereum IDE*. accessed: Jan. 04. 2022. Accessed: Jan. 4, 2022. [Online]. Available: https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

[41] *A Crypto Wallet & Gateway to Blockchain Apps*. Accessed: Dec. 2, 2021. [Online]. Available: https://metamask.io/

[42] *Rinkeby Test Network*. Accessed: Sep. 2, 2021. [Online]. Available: https://rinkeby.etherscan.io/address/0x7a3c9ce1ae1d8ad16422e59e8e0d149af849f0f6

[43] *Instant Multi-Factor Authentication*. Accessed: Jan. 2, 2022. [Online]. Available: https://miracl.com/

ZIA ULLAH received the B.S. degree in computer software engineering from the University of Engineering and Technology, Peshawar, Pakistan. He is currently pursuing the M.S. degree in software engineering with COMSATS University Islamabad, Islamabad, Pakistan. His current research interests include blockchain technology, cryptocurrency, information security, and the Internet of Things (IoT).

BASIT RAZA received the Ph.D. degree in computer science from International Islamic University (IIU), Islamabad, Pakistan, in 2014. He is currently an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad (CUI), Islamabad. He has published several conference and journal papers of international repute. His research interests include data science, data mining, information security, machine learning, and artificial intelligence.

HABIB SHAH received the Ph.D. degree from the Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, in 2013. He is currently an Assistant Professor with the Department of Computer Science, College of Computer Science, King Khalid University, Saudi Arabia. He is also working on three research projects for KKU and KSA. He has successfully published more than 40 articles in various international SCI and Scopus journals and conference proceedings. His research interests include artificial intelligence, learning algorithms, data mining techniques, time series analysis, and numerical optimization. He is a member of an editorial board, a guest editor, and acts as a reviewer for various journals and conferences as well. He has also served as a program committee member and a co-organizer for numerous international conferences/workshops.

SHAHZAD KHAN received the B.S. degree in computer science from Malakand University, and the M.S. degree in computer and communication security from the School of Electrical Engineering and Computer Science (SEECS), National University of Science and Technology (NUST), Islamabad, Pakistan. He is currently pursuing the Ph.D. degree with the Military College of Signals (MCS), NUST. He is also an Assistant Professor with Shaheed BB University. His research interests include applied cryptography and information security.

ABDUL WAHEED received the master's and Ph.D. degrees in computer science from the Department of Information Technology, Hazara University Mansehra, in 2014 and 2021, respectively. He has completed his Ph.D. research from the NetLab-INMC under the School of Electrical and Computer Engineering (ECE), Seoul National University (SNU), South Korea, in 2019, under the HEC Research Program. He is currently a member of the Crypto-Net Research Group, Hazara University. He also works as an Assistant Professor with the Department of Computer Sciences, and the Dean of the Faculty of Engineering and Information Technology (FEIT), Northern University, Nowshera. He has numerous publications in journals and international conferences. His research interests include information security, secure and smart cryptography, heterogeneous communications within the IoT, mobile ad hoc networks (MANETs), wireless sensor networks (WSNs) security, and fuzzy logic-based decision-making theory.