blockchains

MDPI

Article

JudicBlock: Judicial Evidence Preservation Scheme Based on Blockchain Technology

Tapasi Bhattacharjee [1,†,‡], Amalendu Singha Mahapatra [2,*,†,‡], Debashis De [3,‡] and Asmita Chowdhury [4,†,‡]

[1] Department of Information Technology, Techno International New Town, Kolkata 700156, West Bengal, India; dr.tapasi.bhattacharjee@tint.edu.in
[2] Department of Basic Science and Humanities, Techno International New Town, Kolkata 700156, West Bengal, India
[3] Department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, Nadia 741249, West Bengal, India; debashis.de@makautwb.ac.in
[4] Department of Computer Science and Engineering, Techno International New Town, Kolkata 700156, West Bengal, India; asmita.chowdhury.cse.2022@tint.edu.in
* Correspondence: amalendu.singha.mahapatra@tict.edu.in or amalendu007@gmail.com
† Current address: Techno International New Town, Kolkata 700156, West Bengal, India
‡ These authors contributed equally to this work.

Abstract

The electronic judicial evidence preservation systems face various challenges including regulatory control, data exchange, poor credibility, etc. To address these issues, a blockchain-based judicial evidence preservation framework, JudicBlock, is proposed in the present study. It combines the scalability of the Interplanetary File System with the transparency and security of public blockchain. By decentralizing data management and using cryptographic integrity, the system ensures reliable chronological tracking of investigative changes. Unlike traditional approaches, JudicBlock incorporates smart contracts and advanced consensus mechanisms to enforce strict access controls with secure collaboration among the stakeholders. The simulation results show that JudicBlock provides better results over traditional ELR (electronic law records) storage schemes in terms of mining cost, query fetching time, block processing IPFS (Interplanetary file systems) throughput, etc. At a USD 6 mining cost, it appends an average of 23,601 transactions. For 25 blocks, the average query fetching time is 0.852 ms with the cache support of 32 KB. The proposed scheme achieves an average ELR uploading latency improvement of 6.79% over traditional schemes. The results indicate the efficacy of the proposed scheme over the conventional schemes.

Keywords: electronic law records; judicial evidence; blockchain technology; smart contracts; interplanetary file system

Academic Editor: Keke Gai

Received: 19 August 2025
Revised: 22 September 2025
Accepted: 24 September 2025
Published: 26 September 2025

Citation: Bhattacharjee, T.; Mahapatra, A.S.; De, D.; Chowdhury, A. JudicBlock: Judicial Evidence Preservation Scheme Based on Blockchain Technology. *Blockchains* **2025**, *3*, 11. https://doi.org/10.3390/blockchains3040011

1. Introduction

The rise of digitization has significantly transformed the traditional reliance on physical documents into a preference for electronic records across numerous fields. It includes the judicial system as well. Law and governance have embraced this shift by adopting electronic law records (ELRs). These comprise documented testimonies from various individuals, details of accused cases, and critical information on primary suspects related to legal proceedings [1–3]. These trials, which are evidence-based and begin with the filing of a "First Investigation Report" (FIR) by the competent authority (CA), create a chronological order of investigative activities and document updates recorded in ELRs [4–7].

Despite the efficiency improvements offered by ELRs, stringent policies, established norms, and a reduction in manpower within government-appointed investigative agencies have contributed to a substantial backlog of cases awaiting investigation or fair trial proceedings [5,8–10]. The 'National Judicial Data Grid' (NJDG) of India has predicted that by 2030, the proportion of pending cases is expected to increase to 56.77% for impartial case hearings or investigative trials [11]. The projected increase emphasizes the heavy burden on the judicial sector and the centralized structure of proceedings, which, despite the presence of ELRs, leads to processing delays [12–15].

ELRs have enabled the faster transfer of case files, systematically organized by case numbers, among key legal stakeholders. These include petitioners, accused, police officers, lawyers, law enforcement agencies (LEA), and courts [1–3,6]. In addition to saving space compared to traditional paperwork, ELRs address the challenges of manual storage and improve staff efficiency. These also improve public accessibility to records by allowing federal agencies to receive real-time updates during the investigative process [16–19]. The digital form of the ELR records is stored on centralized government servers, which introduces several issues. Some of them are listed below.

- End-User Latency: The vast number of stored cases in digital form results in significant latency when querying cases by case numbers.
- Single-Point Failures: Single-point failures pose a significant risk to centralized servers.
- Security Risks: These servers become attractive targets for security attacks, such as denial-of-service attacks, data alteration, and impersonation of public identities.
- Draft Ambiguities and Authorization Issues: There are inherent challenges related to uncertainties in preliminary drafts, validation through digital timestamps, and inconsistencies in formatting.

Due to the sensitivity and confidentiality of ELRs, centralized storage is not a sustainable long-term option. Storing records on centralized servers with time stamping increases the risk of privacy and security breaches along with the potential for collusion among multiple parties. These could result in tampering with investigative records. In [6,20,21], a distributed ELR server system is proposed as a solution to tackle these challenges and reduce vulnerabilities and alleviate latency concerns for end-user queries. Collaboration attacks persist, posing a threat of manipulation of the investigation results due to significant data exchanges between cloud services, users, and IoT devices [22,23]. ELRs face challenges regarding authorized transparency, the chronological tracking of events, and trust in the sequence of investigative processes. These issues can result in the manipulation of court proceedings, potentially excluding honest parties [16,24,25]. These challenges are represented in Figure 1.

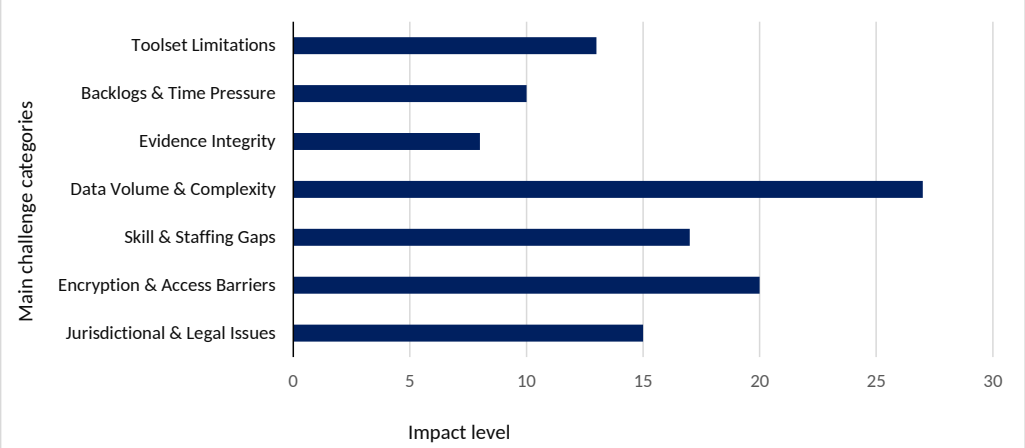

Figure 1. Challenges faced by law-investigating agencies with electronic records [11].

In the realm of legal investigations and judicial processes, blockchain (BC) technology presents a promising approach for managing ELRs through immutability and version control, fostering trust among judicial stakeholders. Using BC, vulnerabilities in current distributed storage systems, such as data loss, disk failures, and trust deficiencies, can be effectively addressed [26–29]. By securing transactions, allowing authorized access and ensuring decentralization and immutability, BC can maintain the authenticity and reliability of digital evidence, making it admissible in court. For legal inquiries, a public BC is preferable because it maximizes trust, security, anonymity, and decentralization [16,25,30]. Public BCs establish a collusion-resistant network. In the context of Proof-of-Work (PoW), miners are tasked with solving complex puzzles and rapidly adjusting block nonces before a new valid block is added. It ensures the integrity of the network. As a result, public BC's block ordering is more fair than chains that are private and consortium-based. These are more vulnerable to collusion between registered nodes. In addition, public BCs maximize security, anonymity, transparency, and resistance to corruption. It also enables the optimization of framework throughput [13,14,26]. To address these issues, the study proposes the use of robust network infrastructures or the adoption of off-chain storage solutions such as the Interplanetary File System (IPFS). In this approach, only meta-information is stored in BC, while indexed references to IPFS are used to ensure that scalability is preserved [18,21,26]. IPFS enables the sharing of large datasets through communication between peers and rapid file retrieval using the Merkle-DAG (directed acyclic graph) method. This method alleviates the burden on BC, allowing a higher volume of transactions to be processed more efficiently [17,22]. The paper reported in [31] outlines blockchain's impact across various forensic fields, especially in IoT, cloud, and storage forensics, where it improves data provenance, access control, and chain-of-custody management. This paper proposes a blockchain integration solution for an IoT forensics framework (PBCIS-IoTF), implemented using Hyperledger Fabric blockchain platforms integrated with a Raspberry Pi IoT device and web access. The design aimed to test blockchain IoT integration technologies to address IoT forensics challenges.

A BC-based digital law evidence scheme, 'NyaYa' is reported in [6]. It is effective for the management of ELRs and future judicial investigations. This scheme proposes a BC-based method for case record storage, processing, retrieval, and updating for judicial investigations. Here, a four-phase scheme is used that highlights the need for chronology, transparency, and trust among judicial stakeholders and provides justice to the appellant in the court of law. The scheme stores case information in BC, with reference to IPFS off-chain, which improves the scalability of chain operations. Inspired by the preceding discussions, this paper presents a framework, 'JudicBlock', for legal inquiry that utilizes a public blockchain to manage and preserve ELRs by leveraging off-chain storage in IPFS.

Both schemes share similar objectives. However, JudicBlock offers a more modular and technically rigorous design. It uses public Ethereum (PoW) with clearly separated smart contract phases—registration, upload, retrieval, and download—alongside a robust access control mechanism using secure hash–key pairs. In contrast, NyaYa's contract structure and key management are less defined. JudicBlock also integrates AES-256 encryption with a detailed mathematical formulation, whereas NyaYa presents encryption at a conceptual level. Its entity-based system design and comprehensive use of Mythril for security analysis further distinguish JudicBlock, offering a better transparency in performance metrics such as gas cost, latency, and throughput. JudicBlock's hybrid off-chain/on-chain model, metadata-only blockchain storage, and detailed visualizations make JudicBlock a more scalable, secure, and transparent solution for judicial ELR preservation.

In addition to addressing scalability and security challenges, the JudicBlock framework has been designed with judicial admissibility and chain-of-custody principles in

mind. Blockchain immutability ensures that evidence records remain tamper-resistant and verifiable, thereby supporting their admissibility in court. Furthermore, AES-256 encryption guarantees confidentiality and controlled access to sensitive legal data, reinforcing compliance with legal evidence-handling standards.

The organization of the paper is as follows. Section 2 reviews current state-of-the-art schemes along with the scope of JudicBlock. Section 3 explains the key terminologies. Section 4 outlines the proposed scheme. Section 5 assesses the effectiveness of JudicBlock relative to traditional methods currently in use. Finally, Section 6 concludes the paper and highlights potential directions for future research.

2. Literature Review and Scope of the Proposed Work

This section provides an overview of various state-of-the-art methods in BC and judicial evidence preservation schemes. In recent years BC-based technologies have been extensively researched in the legal domains. Zhao et al. [32] introduced an effective BC-assisted Conditional Anonymity Privacy-Preserving Public Auditing framework. This framework incorporates an incentive system to motivate and compensate the original data uploader. Research primarily focuses on strengthening privacy protection, especially for individuals sharing sensitive information, by securely protecting their identities while facilitating effective public auditing.

Gong et al. [10] proposed a BC-based technology to improve the sharing of information related to property affected by cases between different departments. They established a multi-departmental alliance chain specifically designed for the transfer of case-related property information. This alliance chain facilitates secure and efficient communication between public security and judicial departments, enhancing collaboration and ensuring that case-involved property information is accurately and transparently shared among the relevant authorities. In order to detect and prevent false checks, Hammi et al. [33] proposed a BC-based solution that allowed checks to be authenticated almost immediately after they were stored. This eliminated the need to initiate the bilateral procedure between the participating agencies. Elokaby [34] explored the impact of these developments on the judicial system in several ways. Cybersecurity laws now offer a legal framework for punishing cybercrime offenders, improve the capacity to investigate and gather digital evidence, and modern legislation also implements effective security measures to protect sensitive data. Huu et al. [35] created an efficient multitask neural learning network model (EMNLNM) that integrates bidirectional long-short-term memory (BiLSTM) with a dilated residual convolutional neural network (DSR-CNN) to predict judicial decisions.

A particular transaction format and blocking mechanisms were introduced to mitigate trust issues related to evidence transfer between various judicial departments, as reported in [36]. Smart Contracts (SCs) were designed specifically for three main processes: sending evidence, retracting evidence, and handling evidence transfer queries. These SCs aimed to streamline and secure the transfer of evidence between departments. However, their work primarily focused on the procedural aspects of evidence transfer and paid less attention to the actual storage and long-term preservation of electronic evidence. Verma et al. [6] proposed a BC-based management scheme, 'Nyaya', to solve the difficulties of keeping electronic legal records in a digital setting. This scheme was designed to address the trust and privacy issues that arise between various legal stakeholders, including plaintiffs, defendants, police, defense lawyers, LEAs, and court judges. The primary focus of their work was on the registration and storage of cases, ensuring the integrity and confidentiality of legal records. However, the scheme did not discuss the aspects of data sharing between case agencies or public dissemination of case information, leaving these areas less explored in their study. In order to increase the trustworthiness of the candidate nodes and improve

the reliability of BC technology, Lingqin et al. [37] proposed a privacy-preserving method for use in open networks. Meanwhile, the authors of [38] designed a BC-based electronic evidence storage system specifically for public security law enforcement. This system meets the functional requirements of electronic evidence in such contexts by emphasizing the authenticity and objectivity of the evidence. To boost system performance, researchers improved the practical Byzantine Fault Tolerance (PBFT) consensus mechanism. This enhancement plays a vital role in preserving the integrity and reliability of the evidence, thus improving the overall efficiency of public security law enforcement.

The scheme reported in [1] preserves judicial evidence electronically. This method leverages IPFS to facilitate reliable and distributed data storage and exchange, successfully addressing the challenge of constrained block space. The proposed system is designed to accommodate various critical scenarios, including storing, sharing, reviewing and publishing case information, as well as uploading and downloading electronic evidence. This comprehensive solution aims to improve the integrity, accessibility, and efficiency of handling electronic judicial evidence.

Most of the schemes reported so far have focused on utilizing BC technology in specific data storage domains. However, limited attention has been paid to the overhead introduced by BC systems and the effect of data storage on block capacity. However, the emergence of BC technology offers innovative solutions for electronic evidence storage. Using consensus algorithms, cryptography, and its chain structure, BC addresses issues related to the low authenticity, admissibility, and verification challenges of electronic evidence. Moreover, when combined with the IPFS, BC can effectively tackle data-sharing and storage cost issues. Indirectly, it facilitates the expansion of the block. Finally, the integration of SCs with IPFS creates a shareable, secure, and efficient solution to preserve judicial evidence.

Scope of the Current Work

Inspired by the discussions above, a BC-based judicial evidence framework, JudicBlock, is proposed in this study. This framework leverages a public BC combined with IPFS for off-chain storage to manage and maintain ELRs. JudicBlock establishes an end-to-end transparent and chronological access mechanism, enabling all registered stakeholders to view and access the data. In this scheme, only the meta-information of ELRs is stored on the public BC, while the actual records are linked via a case number, represented as a hash object corresponding to the record in IPFS. Although the public BC is accessible to everyone, detailed records on IPFS are restricted to registered stakeholders. Access to these records is provided through the IPFS hash object and a private IPFS key. It is granted once the stakeholders are registered and authorized by federal entities.

By combining the IPFS hash object with a private key, registered users can securely access case records, protecting user privacy and ensuring the integrity of law records. Authorized stakeholders, such as lawyers and judges, can update ELRs to incorporate new investigative findings, provided that they adhere to access control policies. This method ensures both transparency and security while leveraging IPFS to mitigate data redundancy issues and reduce the risks associated with disk failures. The proposed framework introduces a secure and decentralized system for the preservation and access control of electronic law records (ELRs) using a combination of blockchain technology and distributed storage off-chain. This study specifically uses the public Ethereum blockchain, operating under a PoW consensus, to record evidence metadata and enforce access control through smart contracts. In parallel, IPFS is used for scalable and fault-tolerant off-chain storage of encrypted legal evidence files. Optimization of data throughput is performed by using IPFS as a lightweight distributed storage layer, thus reducing the chain storage burden and gas fees. The distinctive differences and relative contributions of JudcBlock in

the works reported in [6,7,10,19] are listed in Table 1. The challenges of JudicBlock are also shown in Figure 2 as a block diagram representation.

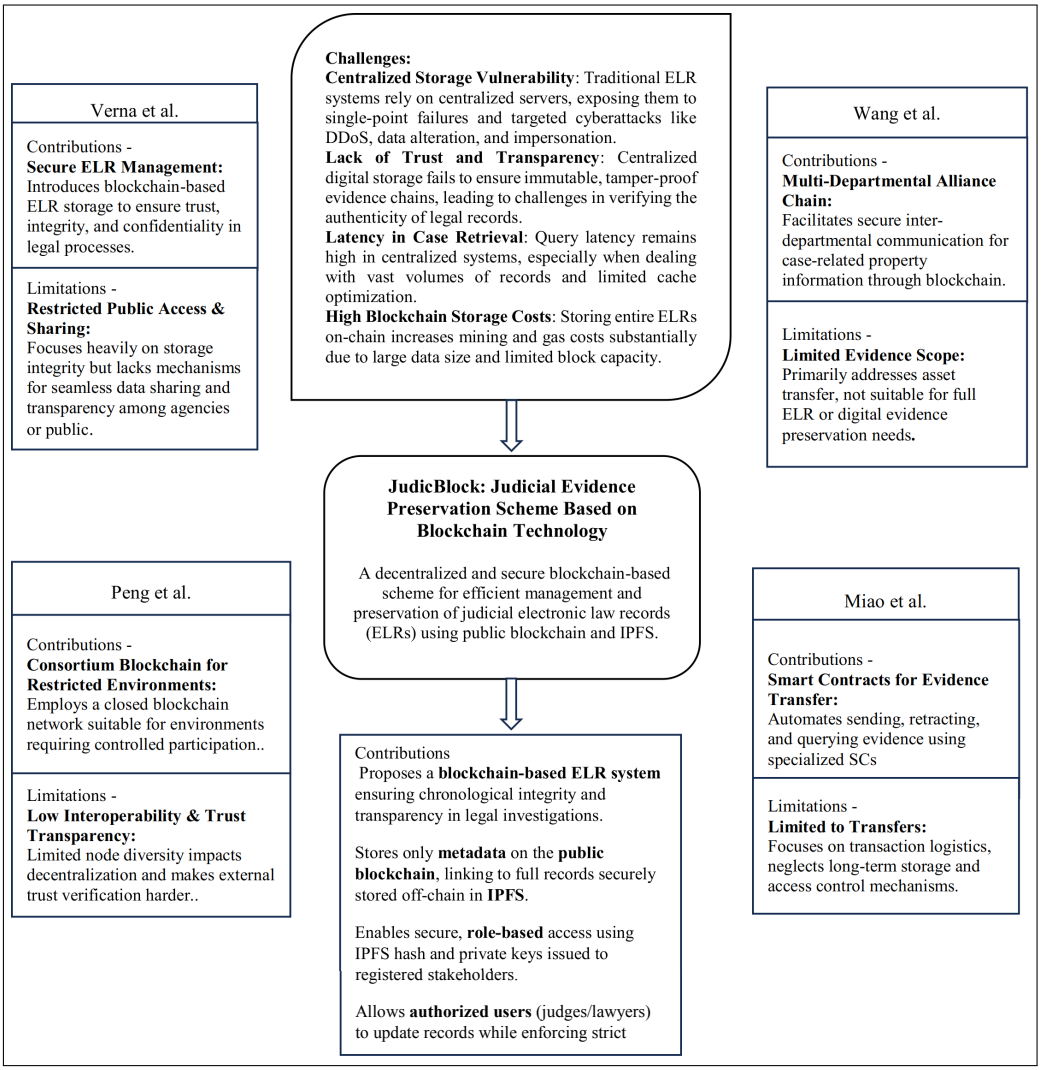

Figure 2. Challenges of schemes proposed by Verma et al. [6], Wang et al. [10], Peng et al. [19], Miao et al. [7] and the contributions of JudicBlock.

Table 1. Comparative table for existing approaches used in ELR using blockchain.

Features	**Schemes Reported in**				
–	**[6]**	**[7]**	**[10]**	**[19]**	**JudicBlock**
Objective	Blockchain-based ELR management with SCs and IPFS	Privacy-preserving authentication for Internet of Medical Things using blockchain	Blockchain-based ELR system for law enforcement integration and evidence sharing	P2P file storage and sharing system using consortium blockchain	Judicial evidence preservation using Blockchain and IPFS with robust SC design
Blockchain Layer	Public Ethereum (PoW), similar SC phases but less modularity	Lightweight authentication protocol; not smart contract-driven or PoW-based	Public blockchain, focus on ELR data transfer, not modular SCs	Consortium blockchain; identity-based access control	Public Ethereum (PoW), detailed SCs for registration, upload, retrieval, download
IPFS Integration	Uses IPFS but with slightly less focus on key control protocols	Does not use IPFS or off-chain architecture	Does not mention off-chain architecture like IPFS	Uses on-chain storage with no IPFS or decentralized off-chain support	Used only for storage, detailed access control via hash and key pairs

Table 1. *Cont.*

Features	Schemes Reported in				
–	[6]	[7]	[10]	[19]	**JudicBlock**
Encryption	Symmetric and asymmetric encryption mentioned, but less depth	Uses Chebyshev chaotic maps and BAN logic for cryptographic modeling	Mentions digital signing but lacks cryptographic detail	Basic identity encryption; no engineering-grade encryption schemes detailed	AES-256 fully explained with mathematical formulation and Galois Field
System Design	Role-based model too, but SC descriptions are more abstract	Protocol-level modeling of IoMT authentication entities; not SC-based	Centralized law enforcement model with blockchain append-only features	Consortium node model with identity verification; no modular SC emphasis	Entity-based model with roles like user, admin, lawyer, judge (Formal SC design outlined)
Smart Contracts	Defined across phases too, but not as explicitly modular	No smart contracts, uses mathematical functions and logic gates for authentication	Basic SC implementation without modular breakdown or gas optimization	Smart contracts used but not modular, lacks gas optimization strategy	4-layered SC model: registration, upload, retrieval, download. Includes Solidity code structure and validation.
Novel Contribution	Chronological ELR updates via hash, SCs for judgment and payments, formal proof structure	Introduces 3-factor authentication with privacy-preserving protocols	System to integrate ELR with blockchain; focus on law enforcement automation	P2P sharing framework over permissioned blockchain	Efficient off-chain/on-chain hybrid, secure access using hash-key control, privacy preservation, metadata-only BC

In summary, the contributions of JudicBlock are summarized below.

- An ELR management and access scheme based on BC technology is proposed to guarantee chronology and transparency in legal investigations.
- The meta-information of ELRs is stored on the public BC and is linked via a case number, which acts as the hash object pointing to the corresponding records in IPFS.
- The public BC is accessible to anyone for viewing. However, only registered stakeholders can retrieve the records stored in IPFS using the IPFS hash object and a private IPFS key, which are provided upon their registration and authorization by federal entities.
- Registered users can access case records using a combination of the IPFS hash object and a private key, ensuring both user privacy and the integrity of law records.
- Judges and lawyers, as part of LEAs, can modify ELR records to include new investigative findings, as long as they follow the appropriate authorization keys and access control policies.
- The system ensures transparency and security while eliminating data redundancy and the risk of disk failures through the use of IPFS.

This section ends with a list of symbols shown in Table 2 to simplify the following of this article.

Table 2. Acronym description.

Abbreviation	Full Form
EMNLNM	Efficient multi-task neural learning network model
BiLSTM	Bidirectional long short-term memory dilated
DSR-CNN	Skip residual convolutional neural network
IPFS	Interplanetary file systems
NJDG	National Judicial Data Grid
PBFT	Practical Byzantine Fault Tolerance

Table 2. *Cont.*

Abbreviation	Full Form
dApps	Decentralized applications
AES-256	Advanced Encryption Standard with a 256-bit key
ELR	Electronic law records
FIR	First Investigation Report
LEA	law enforcement agencies
PoW	Proof-of-Work
PoS	Proof of Stake
DAG	Directed acyclic graph
PoA	Proof of Authority
CID	Content identifier
EQL	Ethereum Query Language
EVM	Ethereum Virtual Machine
BC	Blockchain
SC	Smart Contracts
CA	Competent authority

3. Preliminary Terminologies

A judicial evidence preservation scheme is a systematic approach designed to ensure the secure collection, storage, and management of evidence used in judicial proceedings. Its primary goals include maintaining the integrity of the evidence by keeping it unaltered and tamper-proof from collection to presentation in court, ensuring accessibility for authorized parties such as law enforcement, legal representatives, and the judiciary, and facilitating transparency by providing a clear record of the evidence handling process to build trust and accountability [1,34,39]. In addition, it ensures compliance with legal and regulatory standards for the handling and preservation of evidence. In the context of blockchain technology, this scheme leverages the decentralized and immutable nature of blockchain to improve the security, transparency, and efficiency of evidence management [6,34].

This section presents preliminary terminologies on BC technology, mining and consensus protocols, IPFS, SCs and SHA-256 encryption methodology. This is done to make the article self-explanatory.

3.1. Blockchain (BC) Technology

Blockchain technology transforms data management and security through its decentralized system, which distributes data across a network of computers and ensures resistance to tampering via a peer-to-peer structure. As a distributed ledger, BC maintains a chronological sequence of events or transactions [19,26,33]. Its security is upheld by encryption, consensus mechanisms, and SCs, creating a transparent and immutable record-keeping system. Each block in the ledger is hashed with cryptographic primitives, ensuring the integrity and transparency of the ledger for all authorized stakeholders. BC technology applications are found in a number of industries, including banking, healthcare, education, cloud and edge infrastructure security, and tourism [3,32,40]. Figure 3 shows the BC usage in different sectors. The technology eliminates the need for third-party intermediaries, fostering trust within the ecosystem. Attempts to alter recorded transactions are thwarted by linking the hash of each block to the previous, ensuring that any modification invalidates the entire chain and reveals the source of the alteration [6].

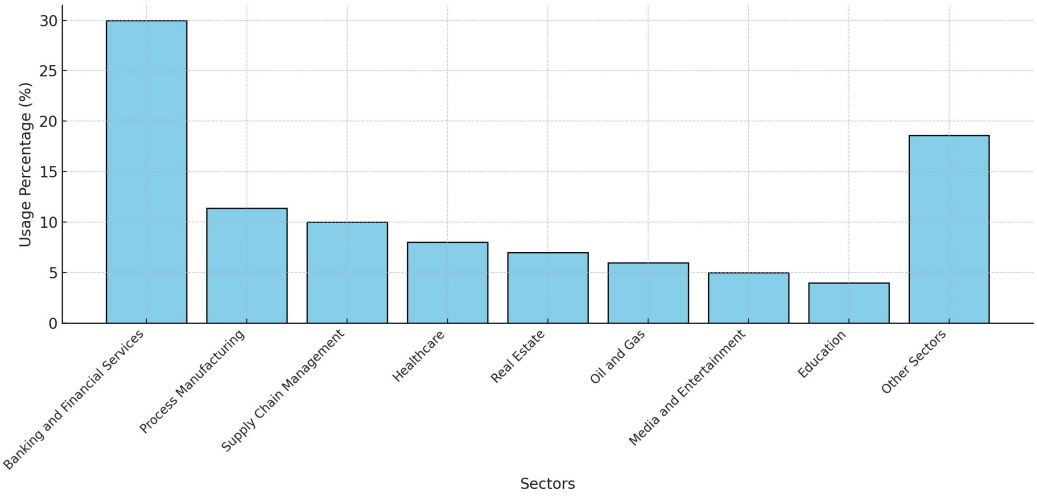

Figure 3. Blockchain usage in different sectors.

3.1.1. Mining and Consensus Protocols

In BC networks, miners are essential to validate transactions and incorporate them into blocks. Miners must find a random nonce value that, when hashed, yields a result below a given target hash to attach a block. This is a computationally challenging task. All miners compete to solve this problem, and the first to do so earns the right to add the block to the BC and is rewarded with incentives. This process requires substantial computational power and storage to run cryptographic algorithms effectively [32,36]. Once a block is added, other nodes in the network verify it using a consensus protocol. The consensus ensures that all distributed nodes agree on the current state of the BC, fostering trust in the system. The consensus mechanism also guarantees that the BC's state is consistent and up-to-date across all nodes. Various consensus protocols exist, such as Proof of Work (PoW), Proof of Stake (PoS), and Proof of Authority (PoA), each suited to different network requirements, including bandwidth, latency, and decentralization levels. The selection of an appropriate consensus protocol depends on the specific needs of the application in question [3,7,38].

In the current JudicBlock prototype, the Ethereum blockchain is used, which employs a PoW consensus algorithm. In this model, miners compete to solve computational puzzles, securing the network by discovering a nonce that results in a hash below a target threshold. PoW was selected for several critical reasons relevant to our system's goals. First, PoW-based Ethereum (before the merge) offers a mature, well-documented development environment leveraging widely used tools like Remix, Meta Mask, and Web3.js which accelerate prototyping and testing smart contracts. Second, PoW is highly secure due to its resistance to Sybil attacks and its high cost of computational manipulation, which is essential in judicial scenarios where data integrity and tamper proofing are paramount. Third, the decentralized nature of PoW minimizes the risk of collusion, unlike PoS or permissioned networks, where the selection of the validator can potentially be influenced. In a judicial context where neutrality and fairness are critical, this transparency and openness is beneficial. Lastly, PoW does not require asset staking, lowering the entry barrier for experimentation and allowing us to simulate a secure, large-scale environment without needing real economic participation.

3.1.2. Smart Contract (SC)

An SC is a digital agreement that runs on its own and has its terms and conditions encoded directly into the code. Unlike traditional contracts, which require human intervention for enforcement, SCs automatically enforce and execute the agreed-upon terms when specific conditions are met. These contracts run on BC platforms, such as Ethereum, ensuring transparency, security, and immutability. Because the code dictates the operations of the contract [5,16]. Eliminates the need for intermediaries, reduces costs, and minimizes the risk of manipulation or fraud. SCs are widely used in diverse domains, including decentralized applications (dApps), supply chain operations, and financial services, making them a fundamental component of BC technology [36]. The operation of a SC is illustrated in Figure 4.

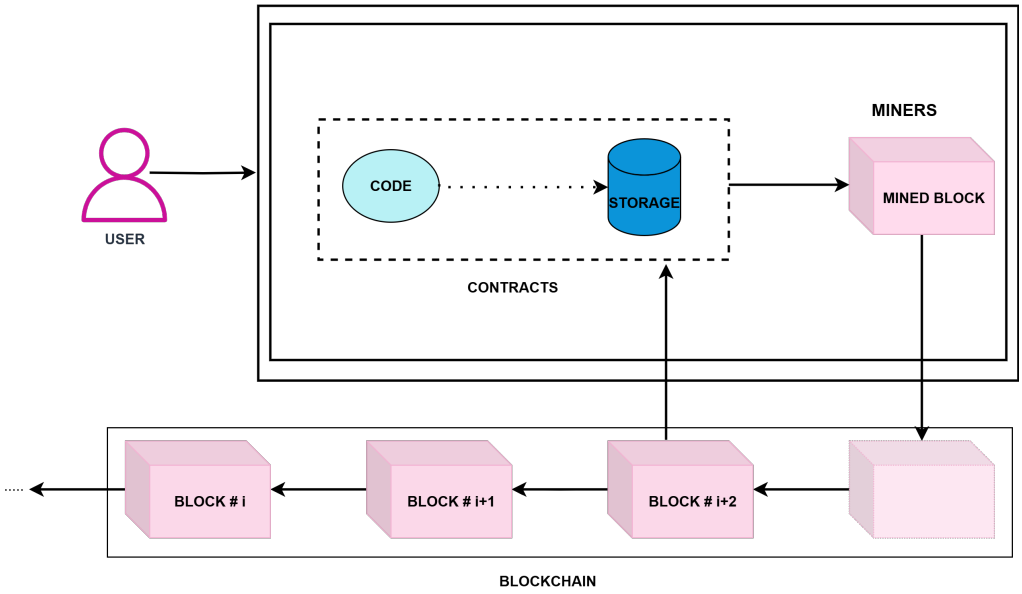

Figure 4. Working of smart contracts.

3.2. Interplanetary File System (IPFS)

IPFS is a decentralized file storage and sharing protocol that enables peer-to-peer data distribution. Unlike traditional centralized systems where data is stored on a single server or group of servers, IPFS distributes and stores files across a network of computers (nodes). Each file or piece of data on IPFS is given a unique cryptographic hash, which acts as a fingerprint for that content. This hash allows data to be retrieved from any node in the network that has a copy of the file, ensuring redundancy and reliability [1,14].

One of the key features of IPFS is its content-addressable nature, which means that files are accessed based on their content rather than their location. This makes it possible to share large amounts of data efficiently without relying on a central authority, and it improves the system's resistance to censorship and data loss. Additionally, IPFS can reduce bandwidth usage since frequently requested files can be cached by multiple nodes, making them available closer to the user. IPFS has various applications, including distributed web hosting, decentralized applications (dApps), and data storage for BC networks, where its decentralized and resilient architecture aligns well with the principles of these technologies [29,37]. In general, IPFS represents a significant step towards a more distributed and secure Internet. The storage schematic of the IPFS structure is represented in Figure 5.

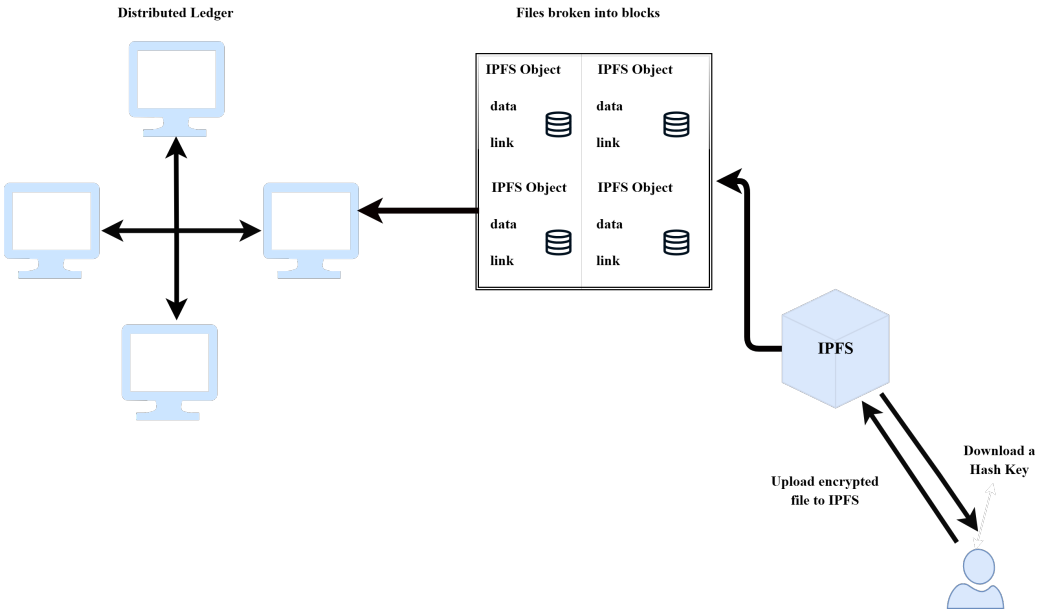

Figure 5. IPFS storage schematic.

3.3. Advanced Encryption Standard with 256-Bit Key (AES-256)

AES-256 is a block cipher that encrypts data using a 256-bit key. It operates on 128-bit blocks of data through a series of transformations such as substitution, permutation, and key mixing [1,25,41]. A brief explanation of AES-256 is given below.

- Data Representation: The 128-bit block is organized into a state matrix 4×4 (S).

$$S = \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \tag{1}$$

  Each element of $S_{i,j}$ is an 8-bit byte.
- Key Expansion: The 256-bit encryption key is expanded to multiple round keys using a key expansion algorithm.
  RotWord : Rotate a 32-bit word $[a_0, a_1, a_2, a_3]$ left by 8 bits.

$$RotWord \quad (a_0, a_1, a_2, a_3) = [a_1, a_2, a_3, a_0] \tag{2}$$

  SubWord: Substitute the bytes using the S-box:

$$S(x) = Ax^{-1} + b \tag{3}$$

  where, A, x^{-1}, b are the constant matrix. multiplicative inverse in GF(2^8) and Constant vector, respectively.
- Encryption Process: AES-256 encryption applies 14 rounds of transformations, with each round comprising four steps. Each of them is briefly explained. Non-linear Byte Substitution: Each byte s_{ij} in the state matrix is replaced using the S-box, which is derived from the Galois field $GF(2^8)$.

$$s'_{i,j} = S(s_{ij}) \tag{4}$$

where x^{-1} represents Byte's multiplicative inverse in $GF(2^8)$ Linear mixing of columns: Each column of the state matrix is treated as a polynomial over $GF(2^8)$ and multiplied by a fixed polynomial.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{5}$$

All arithmetic is performed in $GF(2^8)$, with addition as XOR and multiplication as modular reduction by $x^8 + x^4 + x^3 + x + 1$.
Key Mixing: The state matrix is XORed with the round key.

$$s'_{i,j} = s_{i,j} \oplus k_{i,j} \tag{6}$$

$k_{i,j}$ is the corresponding byte of the round key.

- Decryption Process: The decryption process reverses the encryption process. The process is briefly discussed below. Use the inverse S box for SubBytes.

$$S^{-1}(x) = A^{-1}(x + b^{-1}) \tag{7}$$

Reverse row and column shifts for row reversal and linear mixing of columns. XOR with round keys in reverse order.

AES-256 plays a crucial role in securing data in sectors such as finance, government, and defense by offering strong resistance to cryptanalytic and quantum threats. Its layered encryption process ensures robust protection of 128-bit data blocks, making it a future-ready standard for data confidentiality.

4. JudicBlock: System Model and the Proposed Scheme

This section outlines the system model and introduces the proposed JudicBlock scheme.

4.1. System Model

In this paper, a blockchain-based judicial evidence preservation scheme is proposed that integrates blockchain and IPFS. The block schematic of the model is demonstrated in Figure 6.

The blockchain-based document management system incorporates a comprehensive set of entities. These are indicated as e = eu, ep, eb, es, ea, elea, eul, euj, euv, ela which includes end-users (eu), the platform administrator (ep), the blockchain system (eb), the IPFS storage system (es), and the encryption and decryption system (ea). It also includes lawyers, judges, viewers, and the system administrator. Table 3 lists the notation that is used in the system model of JudicBlock.

The system operates through a multiphased workflow designed to ensure security, transparency, and efficiency. In the User Registration and Login phase, end users interact with the platform to register or log in, with authentication achieved through email credentials and OTP verification, granting access to upload and search functionalities. Behind the scenes, the registered user is checked if it is an authorized user or not. In the foreground, the system makes sure that only authorized users can continue by confirming that the user's ID matches the unique ID of an authorized lawyer or judge. After verifying the caller's identity, the enrollment ID is also verified to avoid invalid input, guaranteeing safe and dependable access control.

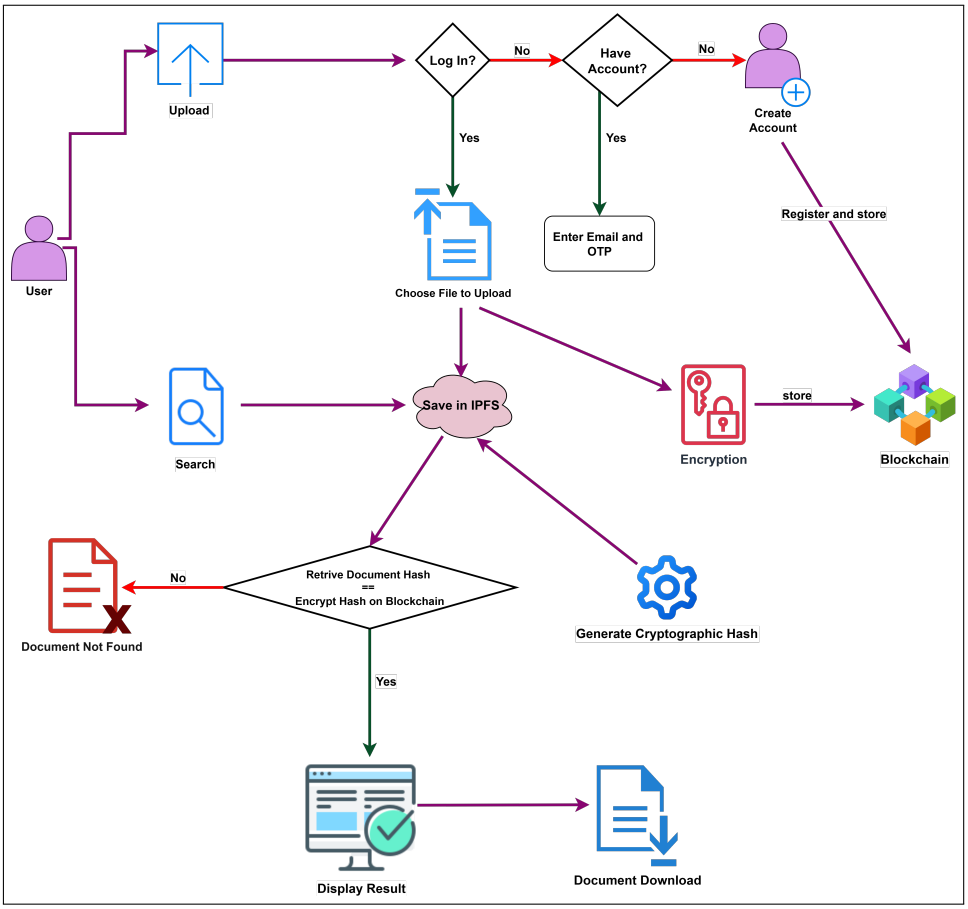

Figure 6. Schematic diagram of JudicBlock.

Table 3. Notations for system model of JudicBlock.

Entity	Notations Used in System Model of JudicBlock
eu	end-users
ep	platform administrator
eb	blockchain system
es	IPFS storage system
ea	encryption and decryption system
eul	lawyers
euj	judges
euv	viewers
ela	system administrator

The Document Upload phase involves hashing and encrypting the uploaded document to generate a unique content identifier (CID) and securely storing it in the IPFS system (es), while cryptographic hash metadata is stored immutably on the blockchain (eb) via smart contracts. In the Document Retrieval and Verification phase, users can query documents by their CID, with authenticity verified by comparing the document hash from IPFS with the encrypted hash on the blockchain. Verified documents can be accessed in the Document Download and Display phase, where decryption through ea ensures authorized access. Finally, the Data Integrity and Transparency phase maintains tamper-proof records of document transactions, including timestamps and identifiers, stored immutably on the blockchain (eb), enabling traceability and fostering user trust. This blockchain-integrated system uses decentralized IPFS storage and robust encryption mechanisms to establish a secure, transparent, and efficient document management solution.

4.2. SC Design

This section introduces a BC-enabled digital management framework designed for judicial documentation. The entity set, denoted as e = eul, euj, euv, ela, encompasses lawyers, judges, viewers and the system administrator, respectively. Notably, the system administrator holds exclusive authority to deploy SCs and incorporate authorized lawyers and judges into the BC network.

JudicBlock operates through a tripartite structure. In the initial phase, user authentication is facilitated via secure login protocols, granting selected users, judges and lawyers, the privilege to securely upload document hashes to the BC. Subsequently, the second phase leverages SCs to efficiently encode and store case-related files and evidence on the BC. Each document is encrypted and securely stored using IPFS, with a unique identifier generated to guarantee data confidentiality and integrity. The final phase focuses on data retrieval and verification, where queries are executed on BC metadata. The authenticity of the document is confirmed by comparing the recalculated hash with the original stored hash, ensuring both the reliability and integrity of the judicial records within the system.

4.2.1. Phase 1: Registration Contract

The access control structure of the framework is designed carefully. User authentication is initially carried out by confirming that the user's address matches an approved lawyer's address or a judge's address. In order to avoid accepting erroneous input, the system verifies the enrollment ID once the caller's identity has been successfully verified. If access is allowed and the enrollment ID is found to be valid, that is, if it does not equal bytes32(0), the function returns a Boolean result of true. On the other hand, the function returns false in the event of an invalid enrollment ID or unwanted access attempts, protecting the process's security and restricting access to authorized organizations. The flow of the contract for the registration of the case is shown in Algorithm 1.

Algorithm 1 Registration Contract

Input: lawyerAddress: Address of an authorized lawyer, judgeAddress: Address of an authorized judge, user: The address of the user requesting access, enrollment: The ID that needs to be validated.
Output: True or False

```
Function validateEnrollment(lawyerAddress, judgeAddress, user, enrollmentID)
Check if the user is authorized (either a lawyer or judge)
if (msg.sender == lawyerAddress) OR (msg.sender == judgeAddress) then
    Validate the enrollment ID
    if enrollmentID != bytes32(0) then
        Grant access to the user and return true
    else
        Enrollment ID is invalid, return false
    end if
else
    Caller is unauthorized, return false
end if
```

4.2.2. Phase 2: File Upload Contract

It defines a document structure containing an IPFS hash and a timestamp. It also uses a mapping to link each document ID (generated as a hash) to its corresponding document. The Contract generates a unique identifier (docId) for each document by concatenating the IPFS hash, the sender's address, and the current timestamp. The generated IPFS hash and the corresponding timestamp are stored within a mapping, ensuring that the references are readily accessible for future retrieval. An event, DocumentAdded, is emitted to log the

successful addition of the document, providing a transparent audit trail. The metadata stored on the Ethereum BC is the IPFS hash (CID) and the timestamp. The complete document is not stored. The necessary references are stored. This is done to retrieve the documents from the IPFS and confirm their authenticity. The flow of this contract is written in Algorithm 2.

Algorithm 2 File Upload Contract

Input: ipfsHash-The content identifier (CID) of the document stored on IPFS
Output: Event DocumentAdded(docId, ipfsHash, currentTimestamp)
1: Function addDocument(ipfsHash)
2: Generate unique document ID (docId) = keccak256(ipfsHash, senderAddress, currentTimestamp)
3: newDocument = Document(ipfsHash, currentTimestamp)
4: documents[docId] = newDocument
5: Grant access to the user and return true
6: Emit DocumentAdded(docId, ipfsHash, currentTimestamp)

4.2.3. Phase 3: File Retrieval Contract

This contract serves as a mechanism to verify the existence of a document by checking the presence of its corresponding IPFS hash within the mapping. If the document is found, the function returns both the IPFS hash and the timestamp; otherwise, it raises an error to indicate the document's absence. The flow of this contract is shown in Algorithm 3.

Algorithm 3 File retrieval contract

Input: docId-The unique identifier of the document
Output: Tuple (ipfsHash, timestamp) or an error message
1: Function getDocument(docId)
2: **if** length(documents[docId].ipfsHash) != 0 **then**
3: Return (documents[docId].ipfsHash, documents[docId].timestamp)
4: **else**
5: Invalid Document
6: **end if**

4.2.4. Phase 4: File Validate and Download Contract

The validateAndDownloadDocument contract adds even more functionality to the document retrieval and verification process. Based on the metadata obtained, this method creates a download URL using the IPFS hash and starts the actual document download. This two-pronged method confirms that a document exists and makes it easier for authorized users to easily access it and obtain the information they need. This contract is depicted in Algorithm 4.

Algorithm 4 File validate and download contract

Input: docId-The unique identifier of the document
Output: Success- "Document successfully downloaded" (if the document exists and is successfully retrieved) or Failure- "Document does not exist or could not be retrieved" (if the document does not exist or the download fails).
1: Function validateAndDownloadDocument(docId)
2: (ipfsHash, timestamp) = getDocument(docId)
3: **if** ipfsHash != NULL **then**
4: downloadURL = "url" + ipfsHash
5: Download the document from downloadURL
6: Return "Success: Document successfully downloaded"
7: **else**
8: Return "Failure: Document does not exist or could not be retrieved"
9: **end if**

5. Performance Evaluations of JudicBlock

This section discusses the performance evaluation of JudicBlock. This evaluation is based on the following parameters, namely mining cost, M_{CST}, associative cache-based query fetching time μ, IPFS bandwidth for processed blocks, B_s. By substituting the distributed storage IPFS for the centralized database, the throughput of the scheme is increased, the issue of block capacity constraint is resolved, and the confirmation time of transaction information is expedited. The influence of query latency caused by the execution of Ethereum Query Language (EQL) tags on cached blocks is further examined by varying the cache size μ. The simulation results are carried out on a server with a dual-core CPU and 4G RAM. The analysis is done in the following parts, namely cost–benefit analysis, scalability analysis, IPFS transmission speed analysis, and throughput analysis.

5.1. Experimental Setup

For the BC setup, JudicBlock has used a Linux Ubuntu LTS v18.04 system, running Node.js with npm version 6.7.1. The hardware specifications include Intel Core i5 processors, 4GB of RAM, and 500GB of external storage. For SC development, Remix Ethereum v0.10.3 is used, integrating MetaMask and web3.js libraries. To formally validate SC, the open source tool Mythril [40] is used. This identifies potential security vulnerabilities such as origin-related issues, reentrancy, order dependence and timestamp dependencies that could be exploited. To evaluate, the selected cache sizes are 16 MB, 32 MB, and 64 MB. These sizes correspond to common cache configurations in commodity servers and are widely adopted in system performance benchmarking. They also allow us to observe scalability trends under increasing cache availability and to analyze query response performance across different cache capacities.

5.2. Simulation Results

The hash keys are taken into account to evaluate the simulation results. They are mapped to the IPFS database and evenly dispersed. The mining cost of block B is evaluated on the basis of the number of transactions per block. The block details comprise 4 bytes for block information, 80 bytes for the complete block header, and 2 bytes for the transaction count. Each entry M_{CST} includes a 4-byte case ID and a 4-byte IPFS hash address that links to the stored file. For 1000 transactions, the size of M_{CST} is approximately 3.7 KB. In the third quarter of 2020, the mining cost for Ethereum data was 13.82 USD/KB. The mining costs are compared with [1,6] and are shown graphically in Figure 7. At a cost of USD 6, JudicBlock can handle 23,601 transactions, while [1] could handle 20,348, and [6] could accommodate 22,456 transactions. This improved efficiency is attributed to the storage of M_{CST} as off-chain IPFS data.

With the cache included, the effect of the query fetching time from IPFS is displayed in Figure 8. The cache size μ has been set to 16 MB, 32 MB, and 64 MB, respectively. It is clear that the EQL time is much reduced with an increase in cache capacity. Due to more associated hits from the locality of reference at 25 blocks and μ as 32 KB, the query time is 0.852 ms.

Figure 9 compares the network bandwidth of JudicBlock with that of traditional schemes [1,6,20]. The bandwidth of Ethereum varies roughly between 200 and 300 Kbps for processing and relaying data on the P2P gossip network. By storing off-chain transaction records, more transactions can be processed in each block, effectively boosting on-chain bandwidth utilization.

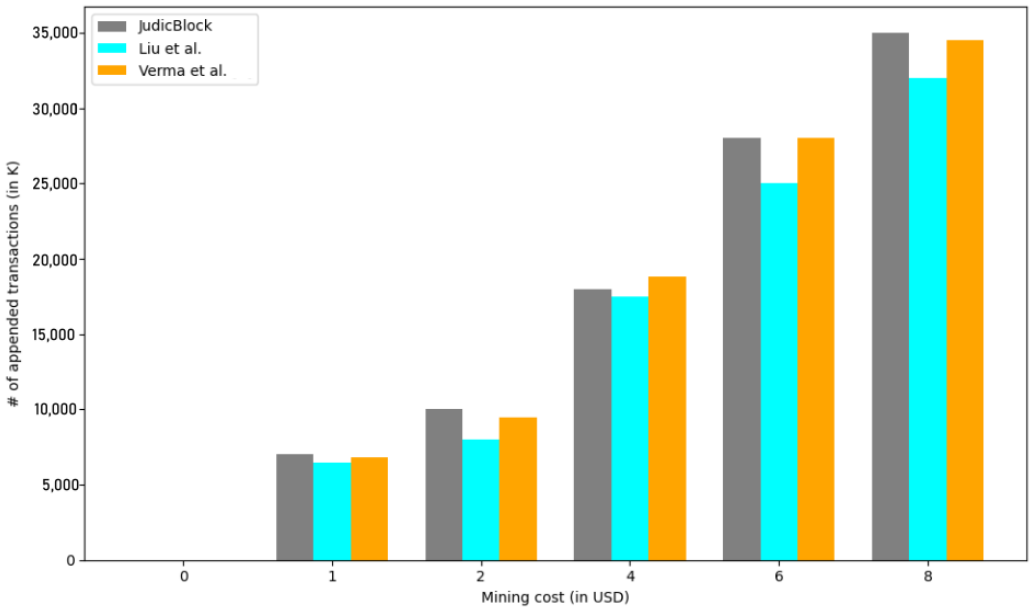

Figure 7. Mining costs, M_{CST} comparison between the schemes proposed by Liu et al. [1], Verma et al. [6] and JudicBlock.

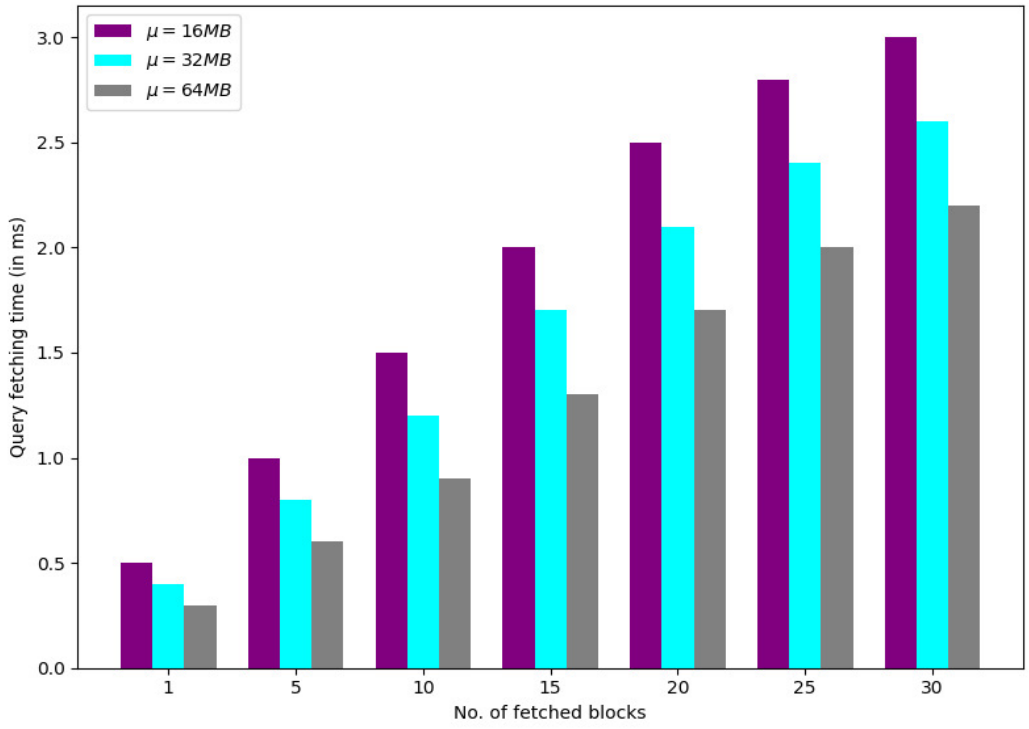

Figure 8. Impact of cache μ to improve cno_k fetch.

The transactional throughput cost of JudicBlock is depicted in Figure 10. Traditional schemes utilize standard databases to store ELRs, where query fetch time, with 64 MB cache proximity, achieves a transactional throughput of 64 MB for a single appended M_{CST} at approximately 100.89 Mbps. In contrast, with IPFS, the throughput is about 90 KBps. This results in an overall throughput of 140.15 Mbps when using IPFS, which is attributed to the storage of meta-information on the BC. This approach enables more transactions to be processed within a given time frame, thereby enhancing overall throughput.

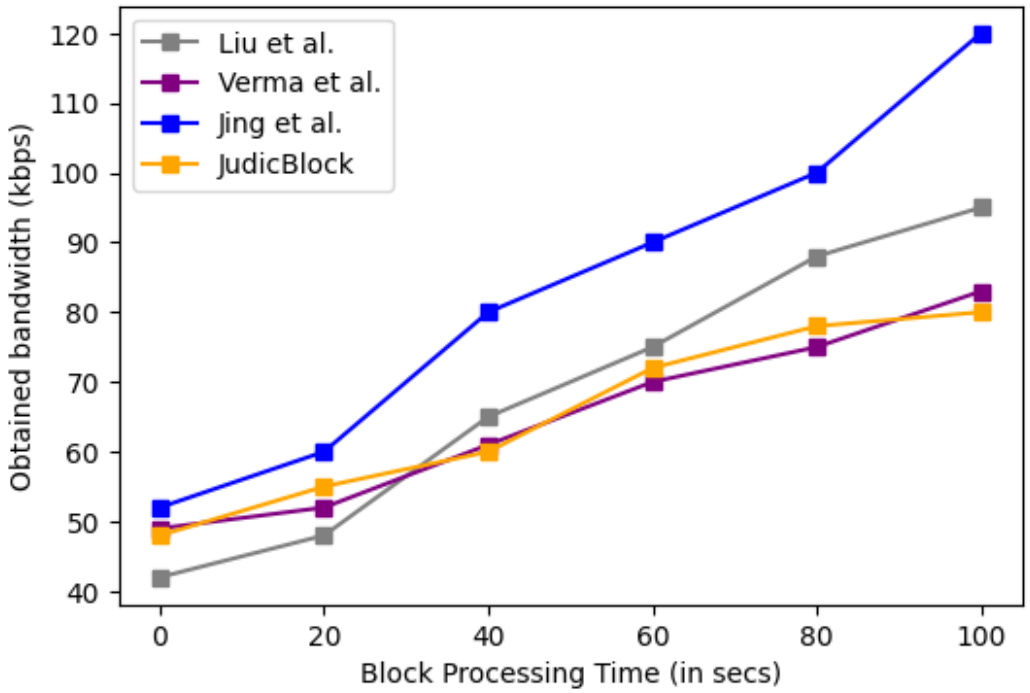

Figure 9. Block processing time vs. obtained bandwidth between the schemes proposed by Liu et al. [1] Verma et al. [6], Jing et al. [20] and JudicBlock.

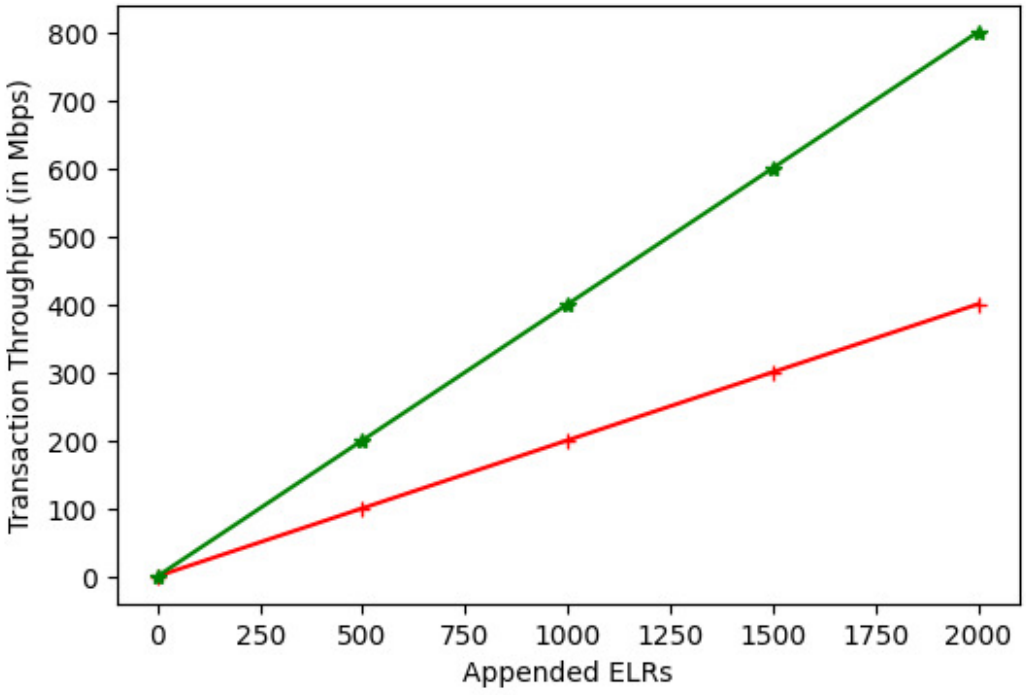

Figure 10. Transaction throughput: IPFS vs. traditional.

Figure 11 illustrates the results of the JudicBlock simulation with respect to execution and transaction costs. The scheme includes four key functionalities, each tracked with a confirmation status and recorded in a transactional ledger on the BC as state functions. In this scheme, eight primary functions (F_1, F_2, . . ., F_8) are defined. Here, F_1 is designated for 'entity registration', with F_2 reflecting the 'transactional output generation'. The function F_3 handles 'case registration'. F_4 represents the 'status confirmation of a registered case'. The function F_5 retrieves 'metainformation from IPFS'. F_6 fetch the 'output status (retrieval

success/failure)'. F_7 serves as the 'payment interface. Finally, F_8 confirms 'payment notifications along with relevant historical transactions'. The tabular representation of the same is listed in Table 4.

Table 4. Execution cost vs. transaction cost of various functions.

Function Label (Contract Name)	Execution Cost (Gas Units)	Transaction Cost (Gas Units)
F1 (Enrollment)	2.5	2.0
F2 (Enrollment)	2.0	1.5
F3 (Storage)	1.8	1.2
F4 (Storage)	1.5	1.1
F5 (Retrieval)	1.3	1.0
F6 (Retrieval)	1.0	0.8
F7 (Download	0.9	0.7
F8 (Download)	0.5	0.4

The following equation illustrates the transactional cost of data storage on BC based on Ethereum [42].

$$T_{CST} = USD6 \times \left(\frac{x \times 20,000}{1,000,000,000}\right) \times 129.34 \quad (8)$$

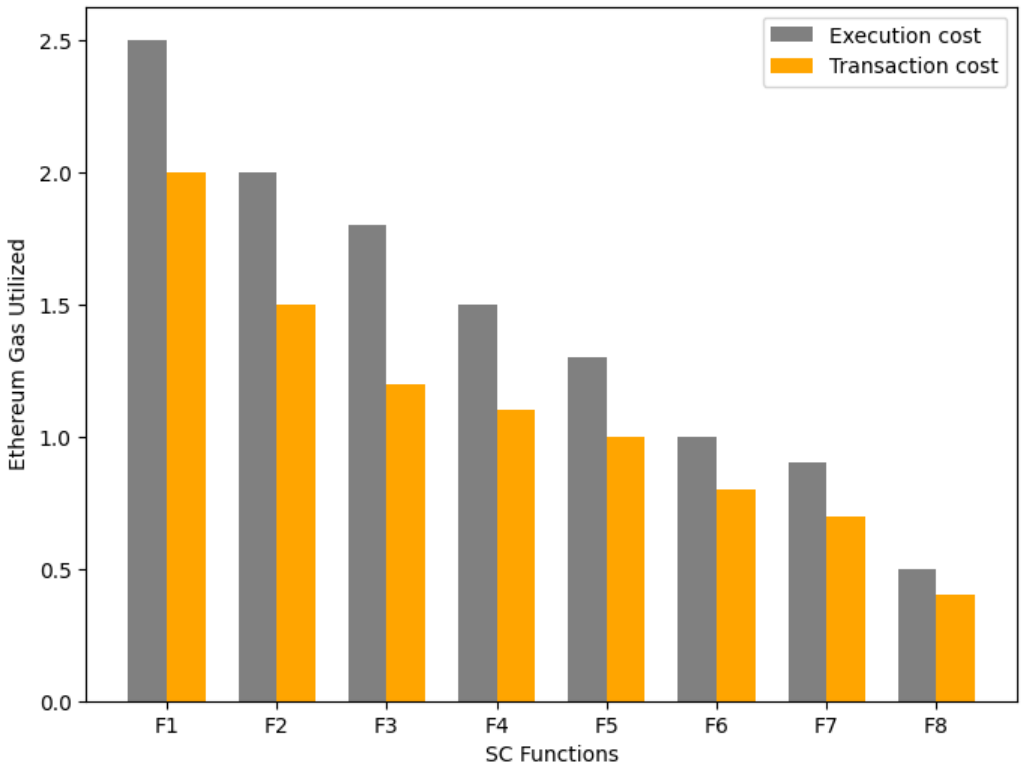

Figure 11. Transaction and execution costs of smart contract functions.

In JudicBlock, data storage is designed to be managed off-chain using IPFS, with retrieval facilitated via a reference hash. This approach helps to minimize the overall transaction and execution costs associated with SC operations. On Ethereum BC, storage costs are determined by word size, where each word occupies 32 KB. Storing a single word requires approximately 20,000 gas. Consequently, storing 5 KB of data requires around $5 \times \frac{2^{10}}{2^5} \times 20000$ or approximately 3,200,000 gas. By averaging these calculations across all functions, the overall transaction and execution costs are determined.

5.3. Functionality of SCs

The SCs are written in the Solidity programming language and compiled on the Ethereum Virtual Machine (EVM) using the Remix IDE. The proposed functionalities are depicted in Figure 12a–d.

In Figure 12a, we introduce the functionality of entity enrollment, where each entity, such as a lawyer or judge, registers with their unique enrollment ID, allowing secure wallet registration and tracking of gas costs associated with each interaction. This access control ensures that only registered entities can interact with the system, safeguarding sensitive legal data. The address for this contract is 0xd9145CCE52D386f254917e481eB44e9943F39138

Figure 12b presents the functionality for case document storage. Here, SCs allow the registration of case documents by any authorized entity, assigning each document a unique IPFS CID for storage.

The address of this contract is 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8. Only metadata, denoted as M_{cno_k}, is stored in the BC, providing a tamper-proof record of the document's existence without revealing its full content. The SC emits an event 'DocumentAdded' for audit purposes, enhancing transparency.

Figure 12c shows the SC functionality for document retrieval, where an entity can search for documents by referencing the IPFS CID linked to the case ID. The SC verifies the authenticity of the document through a hash-matching process, ensuring data integrity, and authorizing access only to verified users. The corresponding SC address 0xf8e81D47203A594245E36C48e151709F0C19fBe8.

Finally, Figure 12d demonstrates the SC's document download capability. When a case file is retrieved, the SC validates its integrity, allowing authorized entities to download the file from IPFS securely. This process ensures that only those with verified access can retrieve sensitive documents, providing a secure and controlled environment for judicial data handling.

The address for this contract is 0xD7ACd2a9FD159E69Bb102A1ca21C9a3e3A5F771B. Through these functionalities, the system ensures the registration, secure storage, reliable retrieval, and controlled access of judicial documents, thus upholding confidentiality and integrity within the BC-based judicial framework.

5.4. Comparative Performance Analysis

JudicBlock creates a stable and scalable environment for the safe handling of documents by combining BC technology with IPFS. The system makes use of sophisticated cryptographic algorithms such as AES-256 and Ethereum's SC capabilities to guarantee the integrity, confidentiality, and immutability of critical legal documents. Important issues like role-based access control, safe peer-to-peer data sharing, and metadata management for decentralized storage are addressed by this design. The framework's capacity is evaluated to reduce the high computational and financial expenses typically associated with BC systems while improving security, scalability, and user accessibility. The system outperforms current systems in terms of consensus processes, platform efficiency, hardware dependencies, energy optimization, and retrieval latency, according to comparative evaluations. JudicBlock offers a safe, open, and user-friendly solution that is suited to the requirements of professional and legal settings, marking a substantial breakthrough in the field of BC-based document management. It is important to create an ecosystem where operational integrity, accountability, and verifiability are the top priority.

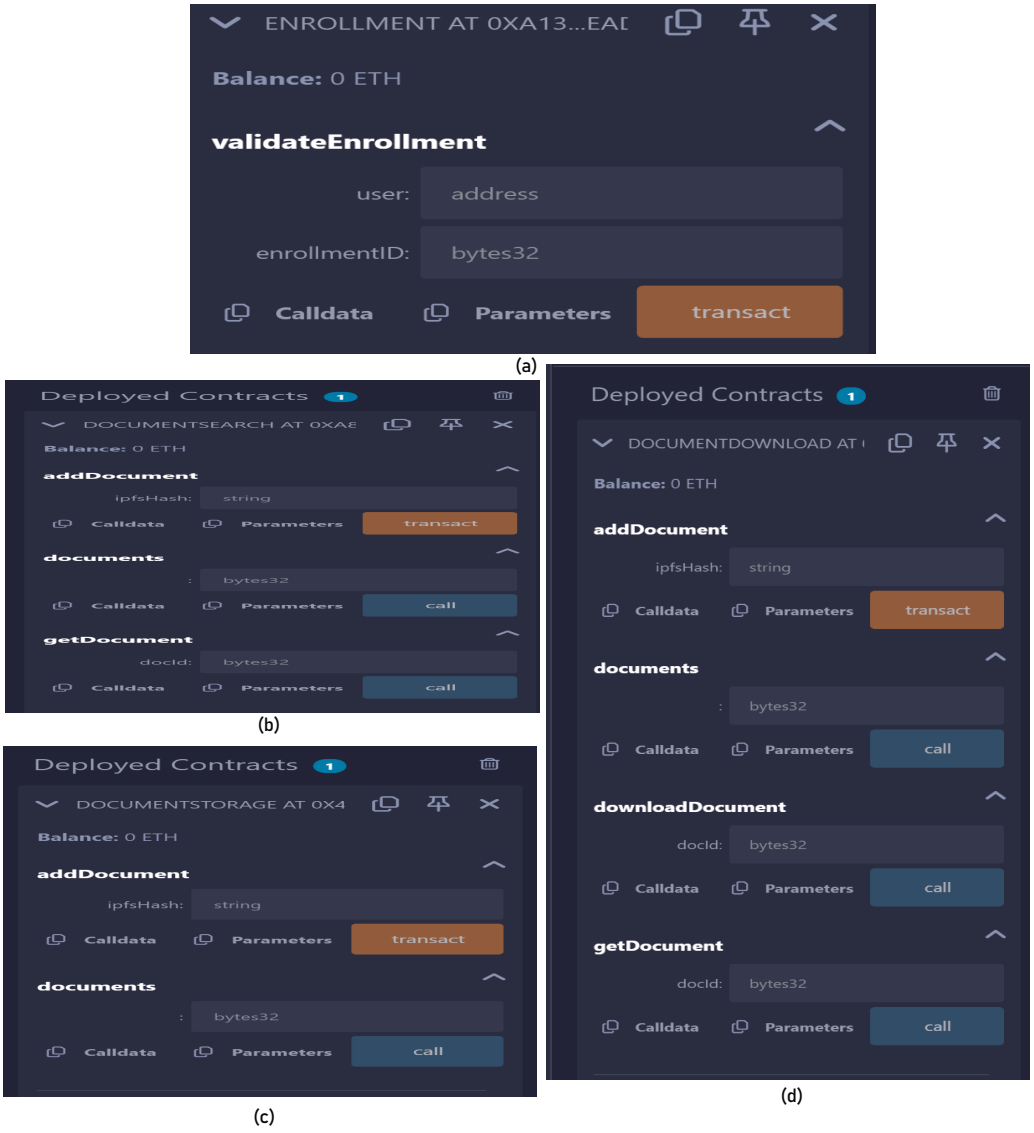

Figure 12. Smart contract functionalities: (**a**) enrollment; (**b**) storage; (**c**) retrieval; (**d**) download.

To evaluate the effectiveness of the proposed scheme, JudicBlock is compared with conventional and state-of-the-art approaches based on a set of selected parameters. Table 5 illustrates this comparative analysis. As shown, the proposed scheme effectively incorporates all the parameters considered and consistently outperforms existing solutions in terms of performance and reliability.

Table 5. Comparative overview of blockchain-based legal/forensic systems.

Paper	Trans-parency	Off-Chain (IPFS)	Security Analysis	Chrono-logy	Signing Latency	Encryp-tion	Perfor-mance	SC Modularity & Gas Opt.	Contribution	Limitation
[36]	×	×	Partial	×	×	×	×	Partial	Introduced SC for trusted storage on Ethereum	No IPFS, performance, or modular contracts
[6]	√	√	Partial	Partial	×	Partial	√	Partial	Full judicial process flow with FIR, updates, SCs	Limited security and cryptographic detail
[10]	×	×	Partial	Partial	Partial	Partial	Partial	Partial	Designed public security ELR system with blockchain	No detailed SC architecture or validation
[33]	Partial	×	Partial	Partial	Partial	Partial	Partial	Partial	Fake check scam detection using blockchain SC	No formal proofs or performance data
[19]	√	Partial	Partial	Partial	Partial	Partial	Partial	Partial	P2P file sharing via consortium blockchain	Lacks off-chain scalability and formal validation
[37]	×	×	Partial	×	×	×	×	×	Proposed a blockchain privacy risk assessment framework	No implementation or experimental metrics
[9]	Partial	Partial	Partial	Partial	Partial	Partial	√	Partial	Integrated blockchain with IoT forensics (Hyperledger)	No signing latency or modular SC details
[7]	Partial	×	√	√	Partial	√	Partial	×	Privacy-preserving authentication for IoMT with formal proofs	No performance metrics or contract modularity
[22]	Partial	√	Partial	√	√	Partial	√	Partial	Built an educational blockchain using Hyperledger	No encryption or security testing
[34]	Partial	×	Partial	Partial	×	×	×	×	Analyzed cybersecurity law impacts in Egyptian judiciary	Purely legal focus, lacks technical system
JuducBlock	√	√	√	√	√	√	√	√	Secure ELR storage with IPFS, SCs, performance, and testing	Focused on ELRs, not broader legal process

6. Conclusions and Scope of Future Works

The proposed work, JudicBlock, is a blockchain-based framework designed to enhance the preservation and management of ELRs for judicial investigations. JudicBlock integrates a public blockchain with off-chain storage through IPFS to create a decentralized, scalable, and secure infrastructure for judicial data management. This architecture ensures transparency and tamper resistance in accessing sensitive legal records, thereby minimizing the risk of collusion and enhancing trust among judicial stakeholders. The performance of the system is evaluated on key metrics, including mining cost, EQL query retrieval latency, and IPFS bandwidth efficiency. In addition, smart contract functionalities are formally verified and successfully implemented to ensure the accuracy and reliability of the scheme. JudicBlock achieves a 6.79% improvement in ELR upload latency, efficiently processed 23,601 transactions at a minimal mining cost of USD 6, and delivered a query response time of 0.852 milliseconds with a 32 KB cache configuration. These results underscore its efficiency and scalability compared to traditional judicial data systems. A comparative analysis (Table 5) highlights that JudicBlock addresses key limitations of existing schemes, including high mining costs, poor scalability, and weak access control mechanisms. By storing only metadata on-chain while preserving encrypted evidence off-chain, JudicBlock balances immutability with privacy, ensuring evidence remains secure yet accessible to authorized stakeholders. Future work will focus on the following:

- Optimizing smart contract operations and exploring interoperability with emerging legal technologies to support broader adoption.
- Extend JudicBlock to other blockchain consensus mechanisms such as Proof-of-Stake for energy efficiency.
- Evaluate real-world judicial case datasets for large-scale deployment.
- Explore regulatory and compliance issues for wider adoption.

Author Contributions: Conceptualization, T.B.; methodology, T.B. and A.S.M.; software, T.B. and A.C.; validation, T.B. and A.S.M.; formal analysis, A.S.M. and T.B.; investigation, A.S.M.; writing–original draft, T.B. and A.S.M.; writing–review and editing, T.B., D.D. and A.S.M.; supervision, T.B., A.S.M. and D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest

References

1. Liu, S.; Zheng, Q. A study of a blockchain-based judicial evidence preservation scheme. *Blockchain Res. Appl.* **2024**, *5*, 100192. [CrossRef]
2. Xiao, N.; Wang, Z.; Sun, X.; Miao, J. A novel blockchain-based digital forensics framework for preserving evidence and enabling investigation in industrial Internet of Things. *Alex. Eng. J.* **2024**, *86*, 631–643.
3. Zhang, H.; Wang, R.; Cai, K. Research on the application and examination of electronic evidence preserved on the blockchain in Chinese copyright judicial practice. *Comput. Law Secur. Rev.* **2024**, *52*, 105891.
4. Ryu, J.H.; Sharma, P.K.; Jo, J.H.; Park, J.H. A blockchain-based decentralized efficient investigation framework for IoT digital forensics. *J. Supercomput.* **2019**, *75*, 4372–4387.
5. Li, S.; Qin, T.; Min, G. Blockchain-Based Digital Forensics Investigation Framework in the Internet of Things and Social Systems. *IEEE Trans. Comput. Soc. Syst.* **2019**, *6*, 1433–1441. [CrossRef]
6. Verma, A.; Bhattacharya, P.; Saraswat, D.; Tanwar, S. NyaYa: Blockchain-based electronic law record management scheme for judicial investigations. *J. Inf. Secur. Appl.* **2021**, *63*, 103025. [CrossRef]
7. Miao, J.; Wang, Z.; Wu, Z.; Ning, X.; Tiwari, P. A blockchain-enabled privacy-preserving authentication management protocol for Internet of Medical Things. *Expert Syst. Appl.* **2024**, *237*, 121329. [CrossRef]

8. Strandberg, K.; Nowdehi, N.; Olovsson, T. A systematic literature review on automotive digital forensics: Challenges, technical solutions and data collection. *IEEE Trans. Intell. Veh.* **2022**, *8*, 1350–1367. [CrossRef]
9. Mbimbi, B.; Murray, D.; Wilson, M. IoT Forensics-Based on the Integration of a Permissioned Blockchain Network. *Blockchains* **2024**, *2*, 482–506.
10. Wang, Q.; Zhang, J.; Liu,W. Design and implementation of a public security law enforcement electronic evidence system based on blockchain technology. *CAAI Trans. Intell. Syst.* **2022**, *17*, 1182–1193.
11. National Judicial Data Grid (District and Taluka Courts of India). Available online: https://njdg.ecourts.gov.in/njdg_v3/ (accessed on 15 May 2025).
12. Xiang, F.; Huaimin, W.; Peichang, S. Proof of Previous Transactions (PoPT): An Efficient Approach to Consensus for JCLedger. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 2415–2424. [CrossRef]
13. Mahdi Salih, K.M.; Ibrahim, N.B. CustodyChainGuardian: Blockchain of Custody Digital Evidence Preservation System. In Proceedings of the 2023 IEEE Asia-Pacific Conference on Geoscience, Electronics and Remote Sensing Technology (AGERS), Surabaya, Indonesia, 19–20 December 2023; pp. 168–175.
14. Srivastava, S.; Kaur, G.; Himank; Singla, S. Implementation of Blockchain and IPFS to safeguard evidentiary data. In Proceedings of the 2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS), Chikkaballapur, India, 18–19 April 2024; Volume 1, pp. 1–6.
15. Rathee, G.; Ahmad, F.; Jaglan, N.; Konstantinou, C. A secure and trusted mechanism for industrial IoT network using blockchain. *IEEE Trans. Ind. Inform.* **2022**, *19*, 1894–1902. [CrossRef]
16. Metcalfe, W. Ethereum, Smart Contracts, DApps. In *Blockchain and Crypto Currency: Building a High Quality Marketplace for Crypto Data*; Yano, M., Dai, C., Masuda, K., Kishimoto, Y., Eds.; Springer: Singapore, 2020; pp. 77–93.
17. Fu, X.; Wang, H.; Shi, P.; Zhang, X. Teegraph: A Blockchain consensus algorithm based on TEE and DAG for data sharing in IoT. *J. Syst. Archit.* **2022**, *122*, 102344. [CrossRef]
18. Ucbas, Y.; Eleyan, A.; Hammoudeh, M.; Alohaly, M. Performance and scalability analysis of ethereum and hyperledger fabric. *IEEE Access* **2023**, *11*, 67156–67167. [CrossRef]
19. Peng, S.; Bao, W.; Liu, H.; Xiao, X.; Shang, J.; Han, L.; Wang, S.; Xie, X.; Xu, Y. A peer-to-peer file storage and sharing system based on consortium blockchain. *Future Gener. Comput. Syst.* **2023**, *141*, 197–204. [CrossRef]
20. Jing, Z.; Cao, C.; Qin, X.; Wu, H. Blockchain Based Certificate Deposit System for Judicial Departments. In *Proceedings of the Innovative Computing Vol 2—Emerging Topics in Future Internet*; Springer : Singapore, 2023; pp. 689–697.
21. Almasian, M.; Shafieinejad, A. Secure cloud file sharing scheme using blockchain and attribute-based encryption. *Comput. Stand. Interfaces* **2024**, *87*, 103745. [CrossRef]
22. Liang, X.; Zhao, Q.; Zhang, Y.; Liu, H.; Zhang, Q. EduChain: A highly available education consortium blockchain platform based on Hyperledger Fabric. *Concurr. Comput. Pract. Exp.* **2023**, *35*, e6330. [CrossRef]
23. Miao, Y.; Huang, Q.; Xiao, M.; Susilo, W. Blockchain Assisted Multi-Copy Provable Data Possession With Faults Localization in Multi-Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3663–3676. [CrossRef]
24. Fu, X.; Ma, H.; Ding, B.; Wang, H.; Shi, P. Subtraction of Hyperledger Fabric: A blockchain-based lightweight storage mechanism for digital evidences. *J. Syst. Archit.* **2024**, *153*, 103182. [CrossRef]
25. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
26. Han, Y.; Zhang, Y.; Vermund, S.H. Blockchain technology for electronic health records. *Int. J. Environ. Res. Public Health* **2022**, *19*, 15577. [CrossRef]
27. Tian, Z.; Li, M.; Qiu, M.; Sun, Y.; Su, S. Block-DEF: A secure digital evidence framework using blockchain. *Inf. Sci.* **2019**, *491*, 151–165. [CrossRef]
28. Kamal, R.; Hemdan, E.E.D.; El-Fishway, N. Forensics chain for evidence preservation system: An evidence preservation forensics framework for internet of things-based smart city security using blockchain. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e7062.
29. Pilares, I.C.A.; Azam, S.; Akbulut, S.; Jonkman, M.; Shanmugam, B. Addressing the Challenges of Electronic Health Records Using Blockchain and IPFS. *Sensors* **2022**, *22*, 4032. [CrossRef] [PubMed]
30. Qi, X.; Zhang, Z.; Jin, C.; Zhou, A. BFT-Store: Storage Partition for Permissioned Blockchain via Erasure Coding. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1926–1929.
31. Igonor, O.S.; Amin, M.B.; Garg, S. The Application of Blockchain Technology in the Field of Digital Forensics: A Literature Review. *Blockchains* **2025**, *3*, 5. [CrossRef]
32. Cebe, M.; Erdin, E.; Akkaya, K.; Aksu, H.; Uluagac, S. Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles. *IEEE Commun. Mag.* **2018**, *56*, 50–57. [CrossRef]
33. Hammi, B.; Zeadally, S.; Adja, Y.C.E.; Giudice, M.D.; Nebhen, J. Blockchain-Based Solution for Detecting and Preventing Fake Check Scams. *IEEE Trans. Eng. Manag.* **2022**, *69*, 3710–3725.

34. Elokaby, M.A. Development of cybersecurity laws in Egypt and its impact on the judicial system (Opportunities and challenges). *Int. Cybersecur. Law Rev.* **2024**, *5*, 563–584. [CrossRef]
35. Huu, P.T.; An, N.T.; Trung, N.N.; Thien, H.N.; Duc, N.S.; Ty, N.T. Judicial decision prediction using an integrated attention based bidirectional long-short term memory and dilated skip residual convolution neural network. *Vis. Comput.* **2024**, *41*, 4199–4220. [CrossRef]
36. Cao, D.; Chen, W. Mechanism of trusted storage in Ethereum based on smart contract. *J. Comput. Appl.* **2019**, *39*, 1073.
37. Lingqin, R.; Changgen, P.; Dequan, X.; Ningbo, W. Privacy leakage risk assessment method based on blockchain technology architecture. *J. Comput. Eng.* **2023**, *49*, 146–153.
38. Guo, J.; Wei, X.; Zhang, Y.; Ma, J.; Gao, H.; Wang, L.; Liu, Z. Antitampering scheme of evidence transfer information in judicial system based on blockchain. *Secur. Commun. Netw.* **2022**, *2022*, 5804109. [CrossRef]
39. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing. 2009. Available online: https://metzdowd.com (accessed on 20 September 2025).
40. Patel, N.S.; Bhattacharya, P.; Patel, S.B.; Tanwar, S.; Kumar, N.; Song, H. Blockchain-Envisioned Trusted Random Oracles for IoT-Enabled Probabilistic Smart Contracts. *IEEE Internet Things J.* **2021**, *8*, 14797–14809. [CrossRef]
41. Guo, H.; Yu, X. A survey on blockchain technology and its security. *Blockchain Res. Appl.* **2022**, *3*, 100067. [CrossRef]
42. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.