Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 2, November 2025, pp. 1065~1086
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i2.pp1065-1086

Blockchain-based handle-research data sharing: a blockchain-based handle system to enhance the privacy and security of research data sharing

Mahamat Ali Hisseine[1], Deji Chen[2], Yang Xiao[1]
[1]College of Electronic and Information Engineering, Tongji University, Shanghai, China
[2]College of Computer and Information Science, China Three Gorges University, Yichang, China

Article Info

Article history:

Received Sep 28, 2024
Revised Aug 7, 2025
Accepted Oct 15, 2025

Keywords:

BLAKE2
Blockchain
Handle system
Research data sharing
Swarm

ABSTRACT

The increasing demand for secure, persistent and interoperable research data (RD) sharing makes traditional systems vulnerable. All research objects should be findable, accessible, interoperable and reusable (FAIR) for machines and people. This paper proposes a novel framework called blockchain-based handle- RD sharing (BHRDS), which integrates the handle system for persistent identifiers (PIDs) with a smart contract for access control and mirror-specific encryption, BLAKE2-based hashing for identity binding and irregularity detection. The system utilizes swarm, a decentralized storage layer, for off-chain data storage while storing only credential metadata and access conditions on-chain. The framework enables secure identity data management, and verifiable credential distribution across multiple mirror sites. We conducted experiments under growing user numbers (10 to 10,000), different encryption key strengths (AES 128, 192, and 256 bits), and blockchain load conditions. Results show that BHRDS achieves high irregularity detection rates (above 97%) and maintains low response times even at scale. In all the test instances, the system performed accurately, demonstrating that BHRDS offers a decentralized data access model that is scalable and aligned with the FAIR principle, making it suitable for next-generation scientific and institutional data sharing.

Corresponding Author:

Deji Chen
College of Computer and Information Science, China Three Gorges University
443002 Yichang, Hubei, China
Email: dejichen@tongji.edu.cn

1. INTRODUCTION

In today's data-driven research landscape, data sharing is increasingly recognized as a vital mechanism for advancing scientific discovery. It links data providers (DPs) and users, and improves the visibility of research [1]. To ensure adequate data sharing, DPs should follow findability, accessibility, interoperability, and reusability (FAIR) principles throughout all stages of data management, a broader topic underpinned by data sharing [2]. Achieving these principles requires not only accessible and well-described data but also infrastructures that ensure secure, persistent, and verifiable access across decentralized environments [3]. Despite its potential, data sharing faces critical challenges, especially concerning privacy, security, trustworthiness, and long-term accessibility [4], [5]. As research becomes increasingly data-intensive, there is a pressing need for secure and persistent infrastructures that enable controlled and verifiable sharing of sensitive data. Over the years, numerous approaches have been proposed to address these challenges. Among these, researchers have explored advanced cryptographic access control schemes.

For instance, Wang *et al*. [6] combines hierarchical identity-based encryption (HIBE) with ciphertext-policy attribute-based encryption (CP-ABE) to provide fine-grained access control, whole delegation, and high performance in a hierarchical attribute-driven encryption scheme (HABE), while Deng *et al*. [7] suggest a framework baptized hybrid attribute-based proxy re-encryption (HAPRE). A semi-trusted proxy can be authorized to convert ciphertexts of an ABE scheme into an IBE scheme without letting the proxy know the underlying messages. These models provide fine-grained policy enforcement, but their dependence on centralized key authorities limits their trustworthiness and resilience in decentralized contexts. Complementing these efforts, cryptographic schemes such as the mutual query data sharing protocol by Lakum and Rao [8], built to resist chosen-ciphertext attacks (CCA) under shared-strong DiffieHellman (SDH), and the ring signature with forward security proposed by Handore *et al*. [9] demonstrate innovative techniques for anonymous and verifiable communication, particularly in cloud environments.

The use of privacy-preserving techniques for data transformation is growing alongside encryption-based controls. Neubauer *et al*. [10] introduce an approach for pseudonymizing medical data that maintains health data apart from the related patient-identifying data, enabling for confidentiality secondary usage of health records in clinical trials without the need for further anonymization processes. Yang *et al*. [11] employ vertical division of medical information to attain an assessment of distinct areas of medical data involving various privacy issues. Similarly, Beaulieu-Jones *et al*. [12] utilized differentially private GANs to produce synthetic clinical data for research, balancing utility and confidentiality. It suggests the synthetic data can be shared with others for hypothesis-generating analyses as if the original trial data were available. These models improve privacy but often lack persistent linkage between data and identifiers, complicating long-term discoverability and interoperability.

The integration of blockchain into data-sharing frameworks has opened new possibilities for decentralization, and trustless access control. Azaria *et al*. [13] developed MedRec, a decentralized record management, contract-based electronic medical records (EMR) sharing platform built on Ethereum. The system gives patients a full, immutable log and easy access to their medical data across providers and treatment sites. Despite its use of blockchain for patient agency and interoperability, MedRec does not address the security of underlying local databases, depending on its existing infrastructure. It also omits built-in solutions for digital rights management (DRM) or contract encryption, and struggle with scalability. Xia *et al*. [14] presents a system based on a permissioned blockchain, the access to the system is limited to those who are invited, and therefore verified by maintaining a log of all users' actions on the blockchain; further accountability is guaranteed. Although the model effectively protects privacy, it relies on centralized roles (issuer, verifier), which introduces potential bottlenecks. In the energy domain, Wang *et al*. [15] introduce secure and auditable private data sharing (SPDS), a blockchain-based framework for trust-free private data computation and data usage tracking. A smart contract is used to specify fine-grained data usage policies. It is a two-phase atomic delivery protocol to ensure the atomicity of data transactions in computing result release and payment. But their approach also relies on a central aggregator to manage the privacy budget. These studies have not resolved the problem of achieving privacy consensus among participants with diverse privacy preferences. Blockchain systems have recently evolved to address storage scalability and cross-chain interaction. Yin *et al*. [16] proposed a hybrid privacy-preserving federated learning method that combines Bayesian differential privacy and function encryption methods. Their system uses a sparse differential gradient to improve the transmission and storage efficiency of federal learning (FL) training. Despite its many potential advantages, FL face the challenge of limited scalability; it requires participants to have a common training task in order to collaborate in training. Therefore, it may be challenging to apply FL to different tasks for different participants. Ma *et al*. [17] presented IoVChain, a blockchain-based framework for data sharing in intelligent transportation. Built on Hyperledger Fabric, the system integrates homomorphic encryption (Paillier) and zero-knowledge proofs (ZKP) to protect sensitive vehicle data. But the system has a limited storage and the experimental results are limited to a simulated environment with controlled scenarios and do not explore diverse conditions or testing scalability. Chang *et al*. [18] introduce SynergyChain, a multichain framework to enable reliable data sharing with controllable data access. By aggregating the data from multiple blockchains and reorganizing it in SynergyChain, we can achieve data reliability with the verification. Meanwhile, SynergyChain provides hierarchical access control based on smart contracts, making access control automated and credible. However, existing multiorganizational data processing methods lack consideration for the segregation and protection of private data. Xie *et al*. [19] integrates an Intel SGX-based distributed storage system (SDSS) to secure off-chain data; besides, an incentive mechanism is developed to facilitate the whole system. The trusted computing base still depends on a key management center and Intel SGX, which likely introduces centralization risks. Additionally, the scalability under various workloads and the efficiency of SGX across high-frequency transactions require experiments. Wang *et al*. [20] introduced HSHB, a hybrid blockchain, a health data-sharing architecture built on hybrid blockchain; Developed an access control strategy based on entity identity and access purpose

called health data access control (HDAC). Data is encrypted and stored off-chain, with on-chain references and metadata, so it's traceable and accurate. While this model effectively blends blockchain decentralization with access control and scalability, it has several caveats. The scheme's reliance on semi-honest cloud servers and designated administrators (AD) creates centralized trust points, despite the decentralized intention.

Although all of these systems provide many security and performance benefits, they lack persistent identifier (PID) support, which results in poor data findability and long-term traceability over time. As a result, organizations may struggle to efficiently locate and track data over extended periods, potentially leading to inefficiencies and challenges in maintaining data integrity. This limitation underscores the need for enhanced systems that support PIDs to ensure robust data management.

Several efforts have been made to overcome the disconnection between persistent identification and secure management. Notably, Guo *et al*. [21] developed a federated rights management system leveraging the handle system for PID resolution across publishing repositories. Weigel *et al*. [22] presented a conceptual framework employing abstract data type (ADT) and the handle system. The framework facilitates cross-disciplinary adoption and infrastructure integration and also ensures provenance information preservation for a long time. Although the trade-off between a graph-based approach and the ability to optimize performance on rigid structures is mentioned, it has not been thoroughly examined. Harvey *et al*. [23] suggested splitting high-data-density journal articles into narrative and data components with persistent digital object identifiers. Their technique addresses data loss and fragmentation and facilitates the retrieval of scientific data. However, the system is based on centralized storage, which may pose risks to data security and privacy. All data can be negatively affected by a system crash. In order to control system risks, all databases should be backed up and access restricted. Quick *et al*. [24] proposed the enhanced robust persistent identification of data (E-RPID) model to support FAIR data access by integrating PIDs into a unified infrastructure. Similarly, Lannom *et al*. [25] studied the constraints and feasibility of aggregated geographically scattered and diverse biological and geological specimen data. It also studied the DOA data model and components to overcome such hurdles and remain FAIR. The data is carefully stored for eventual reuse and can be referenced via a PID. However, in these systems, access to the data may be restricted due to concerns about privacy or private information. As a result, data would be limited. There will be a lot of difficulty when it comes to collaboration and reproduction.

The handle system has long been a foundational infrastructure for issuing and resolving globally unique identifiers, including DOIs in scholarly publishing and research datasets. While it is foundational in aligning with FAIR data principles, ensuring that data remains Findable, Accessible, Interoperable, and Reusable, it ensures persistence and global resolvability. However, it lacks built-in mechanisms for access control, auditability, cryptographic integrity, and decentralized consistency. As a result, systems relying solely on traditional PID infrastructures often struggle with mirror inconsistencies, unauthorized identifier updates, and a lack of verifiable provenance, gaps that become more severe in distributed research environments. Neither E-RPID nor federated handle systems currently offer mechanisms to directly bind access policies, hash verifications, or smart contract logic to identifiers. This evaluation reveals a key limitation across existing solutions: no single model integrates globally resolvable PIDs with blockchain-based access control, decentralized storage, and verifiable data integrity in an end-to-end, FAIR-compliant architecture. We propose blockchain-based handle system for RD sharing (BHRDS) to fill this gap. This novel framework combines the handle system for issuing and resolving globally unique PIDs, a blockchain trust layer for immutable access policy enforcement, swarm for decentralized, scalable storage of research datasets, and BLAKE2 hashing for lightweight cryptographic integrity verification. Unlike previous methods, BHRDS anchors handle system identifiers directly onto blockchain records, enabling consistent tamper-evident, auditable, and durable data resolution across mirror sites. Smart contracts ensure verifiable and enforceable access rights without centralized authorities. Swarm integration decouples data content from control logic, enabling scalable off-chain storage with on-chain pointers. The design realizes an end-to-end solution for secure, FAIR-compliant data sharing across institutional borders.

The main contributions of this paper are summarized as follows,

- We propose a novel a blockchain-based handle system secure management architecture, BHRDS, the secure storage framework for data based on the blockchain and PIDs can effectively solve management and security problems faced by data storage. Using the handle system to provide consistent and permanent access to research material, make retrieval and citation easier.
- We developed a robust, efficient, and secure blockchain-based BLAKE2 that effectively addresses mirror site vulnerabilities. By integrating blockchain, the system ensures that data integrity and authenticity are maintained, significantly reducing the risk of data tampering and unauthorized access.
- We introduced a system integrating management capabilities of the handle system and swarm to tackle the scalability issues. Swarm enhances the system's ability to handle large datasets by providing a

decentralized storage solution, which reduces the load on the blockchain and improves overall efficiency. This approach alleviates the burden on the blockchain itself for storing data. Consequently, it ensures that blockchain can focus on maintaining its security and integrity functions while leveraging the scalability and efficiency advantages offered by the handle system. Swarm is decentralized and distributed, and so it's also always up, making it stable and reliable. Analysis and evaluations prove that our design is effective for data sharing scenario. Simulations prove that the algorithm we designed could obtain an efficient response time and high detection of irregular activities.

The rest of this study is structured as follow, section 2 gives an overview of handle system and related work. In section 3, the model is constructed, and the experiment is put forward for evaluation. Section 4 presents the verification results and discussion of model performance. Finally, section 5 presented the conclusion, limitations, and recommendations for further work.

2. HANDLE SYSTEM OVERVIEW

The handle system offers a robust framework for assigning, managing, and resolving PIDs, known as "handles," for digital objects and other online resources through a global handle service. This system guarantees the uniqueness of handles and their longevity. It represents the identifier/resolution aspect of the digital object architecture (DOA), which extends the internet's capabilities to encompass broader information management beyond merely transmitting digital data from one location to another [3]. A handle is essentially an identifier linked to one or more fields of typed data. Unlike other computing strategies, handles in the handle system are opaque and exclusively associated with the metadata of the resource [26]. This means that updates to the metadata do not invalidate the handles. The system was developed to address the limitations of existing Internet resolution deployments, particularly the domain name system (DNS), in managing identifiers [27]. The handle system integrates DNS and URLs for identifiers, providing a resolution method that can function without relying on DNS and URLs, if desired [28]. Like other handles used in computing, those in the handle system are opaque, containing no information about the underlying resource and being tied only to the metadata [29]. Therefore, changes to the metadata do not invalidate the handles.

The PID as pointer plays a prominent role in the abstraction process as well as in the FAIR principles. FAIR principles emphasize machine-actionability (i.e., the capacity of computational systems to find, access, interoperate, and reuse data with none or minimal human intervention) because humans increasingly rely on computational support to deal with data as a result of the increase in the volume, complexity, and rate of production of data. For data management or data sharing usually digital content related or community specific information, often in a finer granularity and often in a tight connection to the reference, is much more important than bibliographic information. Therefore, there is a need for other governance structures for Handles to ensure reliable PID services with a much higher flexibility in PID usage and policies [30].

2.1. The management of handle system

In 2005, the system participated in an upcoming network study, becoming a crucial component of the global environment for network innovations (GENI) project's digital item registry. The DONA Foundation was established in 2014 to develop and oversee the DOA, with its headquarters in Geneva, Switzerland. By 2015, a multi-primary administrator (MPA) system was implemented, wherein each MPA represents a country and manages its segment, known as the global handle registry (GHR) [31]. The global management organization of the handle system has authorized and certified ten global root nodes. MPAs in various regions operate independently, implementing handle management policies and interacting with individuals and organizations seeking identifier and resolution authorization.

2.2. Handle system service architecture

The handle system employs a hierarchical service model. At the top level is a single Handle service, known as the GHR. Below this are all other handle services, referred to as local handle services (LHS) [32]. Figure 1 illustrates the architecture of the handle system. Each LHS can consist of multiple servers (e.g., Site 1, Site 2, ... Site n), and each server can have several mirror servers (e.g., Server 1, Server 2, ... Server n). Each handle service can maintain its own hierarchical structure of handle services. The GHR includes a listing for every handle prefix. LHS can be managed by individual organizations with administrative responsibilities for handles within their GHR. Each handle comprises two parts: the prefix and the suffix. The prefix is known as the naming authority, and the suffix is referred to as the local name. These two components are separated by the American standard code for information interchange (ASCII) symbol "/" (handle = Handle naming authority / handle local name).

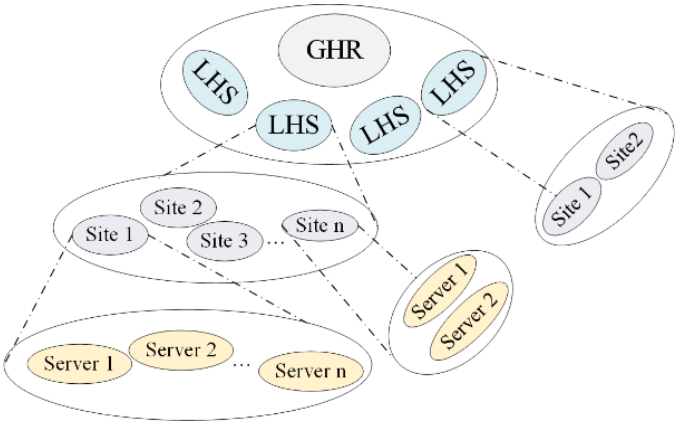

Figure 1. Handle system architecture

3. PROPOSED METHOD

3.1. Method overview

The model incorporates smart contract and the handle system to define a data sharing pact making easier for DP and data customer (DC) to propagate responsible data sharing and understand each other's needs. The process is illustrated in Figure 2.

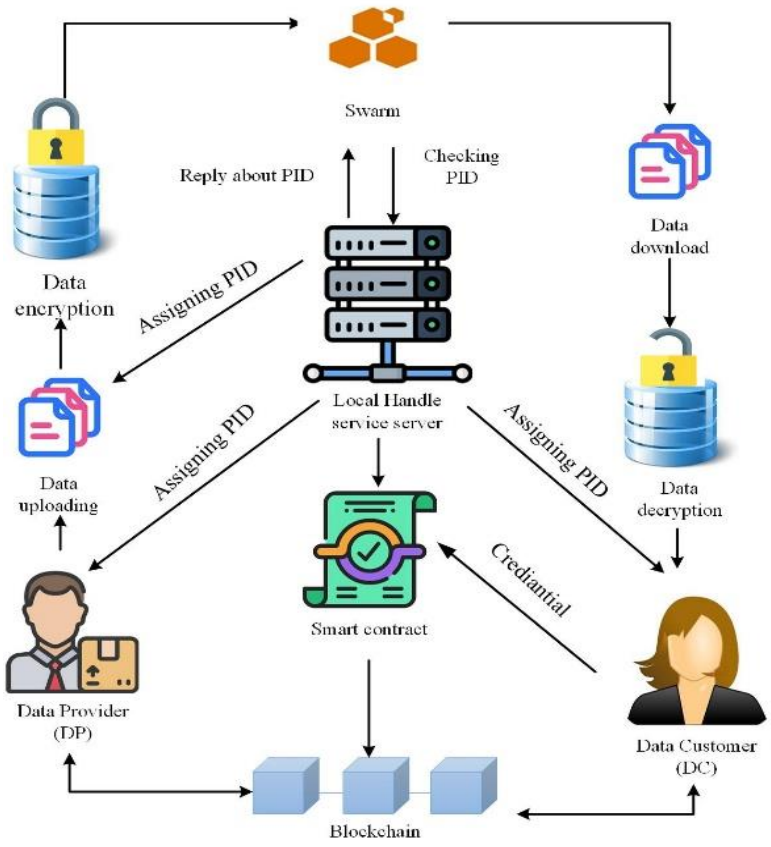

Figure 2. Working process of proposed model

The handle system provides PIDs for research datasets, making them easily citable and accessible and also give identifiers for involved people. In addition, data security method is developed in RD for secure transmission process. The Ethereum swarm, a regulator blockchain (RB), data of the customer, smart contract, and data owner entities are considered in this method. Additionally, DP comprises data protection

scheme in order to secure the RD, where secured RD is transmitted to swarm. The RB is used to record all the operations related to data sharing, including the addition of new data, updates to existing data, and the access control changes. A smart contract is deployed on the RB access control policies. In these contracts, the access control policies of the data are highlighted about read, modify, and delete operations. For any type of request arising due to usage of data from a DC, the smart contract verifies the request against predefined access policies and allows or rejects access accordingly. The RB is utilized to check the integrity of the data that has been stored off-chain in swarm. The cryptographic hashes of data are stored on-chain, and this ensures that data has not been tampered with. During data recovery, a hash of recovered data can be matched against the one stored on-chain as a means of ensuring integrity. The model mainly includes the following stages, setup, assigning PID with handle server PID encoded, customer registration, access credential generation, control, and decoding. DP setup the stage through encoding method of root (PID) and integrating the identifier into RB. The root PID is a prime identifier for operating various codes or encoding the content. This root PID forms an important step in the handle system to ensure secure operations regarding the creation of framework PIDs, data encryption, and secure communications between the components. The smart contract is settled in the RB based on DP. And employed to record the encoded codes contents. During setup, DC make send demand to DP. DP encode then start the process by send the content to the swarm framework, also an encoded data packet in RB. Then, DC copy the content from swarm and decode it.

3.2. Model design

3.2.1. Model initialization

The BHRDS framework begins by establishing a secure and verifiable identity structure for each mirror site. This step is essential for resolving one of the main problems identified in traditional PID-based systems: inconsistencies in mirror site replication and the lack of verifiable identity binding. To address this, we introduce a two-layered persistent identification scheme consisting of a framework PID and a root PID, with both cryptographically bound and anchored in the RB. Figure 3 illustrates a high-level sequence of interactions in the BHRDS framework. It presents the core communication flow between the five primary components: the DC, DP, handle system, Blockchain smart contract (BC), and swarm decentralized storage. The process begins with system initialization and PID setup, followed by user registration, data encryption and upload, access credential generation, and secure data retrieval. This diagram provides a concise overview of the architecture and logic, while full methodological details—including cryptographic formulations, smart contract mechanisms, and credential handling-are thoroughly described in the subsequent sections of the methodology.

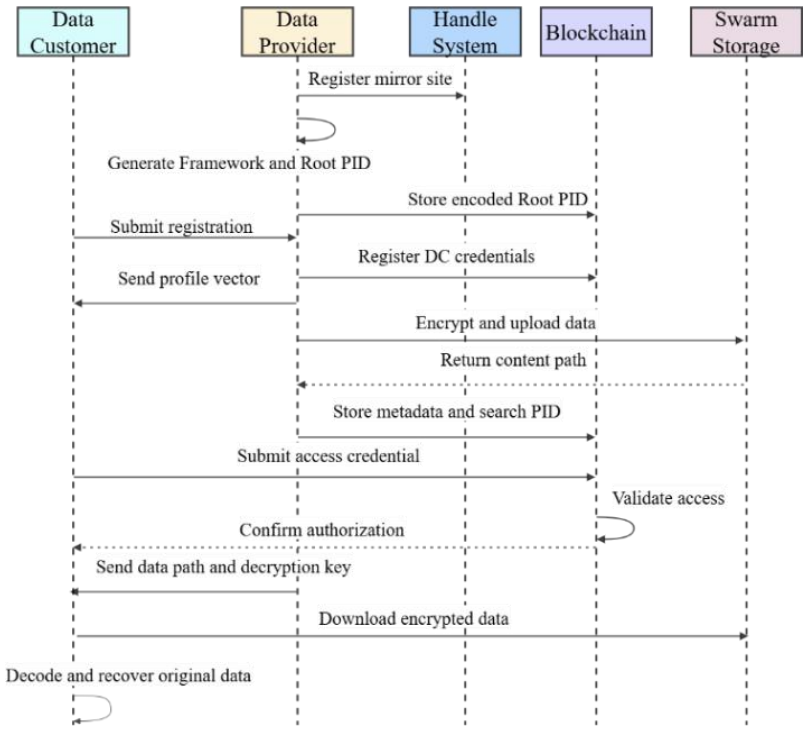

Figure 3. Sequence diagram of the BHRDS secure data sharing process across system components

– Step 1: Generating the framework PID

Each DP initiates the process by generating a framework persistent identifier F_{PID} using a cryptographic hash function,

$$F_{PID} = BLAKE2(M_r \cdot \Theta \cdot M_{id}) \quad (1)$$

where BLAKE2 is a cryptographic hash function selected for its high speed and strong security properties; M_r is a random matrix used to introduce entropy; Θ is a configurable matrix of system-wide security parameters (e.g., time-based seeds, hash chaining rules); and M_{id} is a mirror site identifier matrix unique to each node. We use the BLAKE2 hash function due to its superior performance compared to SHA-2 and SHA-3, along with its resistance to length extension and collision attacks. Hence, F_{PID} will be unique for every mirror site and therefore the inconsistencies across the mirror sites are handled. It is both cryptographically secure and computationally efficient, allowing for practical large-scale deployment. The use of matrix multiplication before hashing enables fine-grained entropy mixing across structural and contextual parameters unique to each mirror, increasing the robustness of identifier uniqueness. F_{PID} is globally resolvable and registered in the handle system. This ensures the "Findable" and "Reusable" aspects of FAIR are satisfied, while the cryptographic construction ensures that the PID cannot be reverse-engineered or fraudulently duplicated.

– Step 2: Encoding and blockchain commitment

The Root PID Ξ is then encoded using a pre-defined algorithm Alg_{enc}, selected to satisfy system constraints such as payload size and smart contract compatibility, given as,

$$\Xi_{enc} = Alg_{enc}(\Xi \times \beta \times M_k) \quad (2)$$

this encoding step generates the final version of the PID ready for on-chain commitment. This process ensures a secure for PID management to each mirror site. The DP implants the encoded Ξ into the RB. The RB receives the encoded Ξ and keep it with the data Δ. The regulator data performs a secure encoding operation with Ξ_{enc} and also records it for further processing. Storing Ξ on-chain ensures that the identifier is tamper-proof, verifiable, and traceable, directly addressing the issue of non-transparent PID management and potential mirror site drift. Blockchain storage guarantees immutability and decentralization, providing persistent provenance information that is critical for reproducibility and trust.

– Step 3: Registration of DC

Furthermore, the DC transfers registration request through the handle server and obtain a persistent identifier (PID) for the customer as Δ_{PID}(data customer identifier) and also a code as Δ_{code}(Data customer code) of DC and sends them to DP. DP checks DC PID and code. The information is saved with a smart contract under $\Delta_{PID'}$ and $\Delta_{code'}$. An access code $\Delta_{a_{code}}$ is delivered to DC by DP. DC receives and records the access code under $\Delta_{a_{code'}}$ and then returns it to the DP after checking it. DP authenticates $\Delta_{a_{code}}$ and distributes a profile vector Λ to DC. This profile is used to define the rules in the BHRDS framework.

3.2.2. Identity verification and management

In the BHRDS framework, identity management is a critical security layer that ensures only authorized and uniquely verified users can participate in data access operations. This step addresses a key limitation in existing systems-namely, the lack of unique identity binding per user and per mirror site, which leads to spoofing, unauthorized access, and inconsistencies across distributed networks. To overcome this, the DP performs a two-stage identity encoding procedure involving a profile vector Λ, the Root PID Ξ, and the mirror site identifier M_{id}. These values are used to compute a unique cryptographic parameter ϕ, which serves as the basis for generating the private code p_c for each DC.

– Step 4: Computation of the identity parameter $\boldsymbol{\phi}$

The parameter ϕ is generated by performing a cryptographic operation between the Root PID Ξ and BLAKE2 of the matrix multiplication of the profile vector Λ, random integer ν, and the mirror site identifier M_{id}. This will make ϕ unique for each mirror site and the profile vector. The parameter ϕ is denoted as,

$$\phi = \Xi + BLAKE2(\Lambda \cdot \nu \cdot M_{id}) \quad (3)$$

the use of BLAKE2 here adds non-invertibility and ensures forward secrecy; even if one ϕ is compromised, others remain secure due to ν. This multi-parameter dependency makes impersonation practically infeasible.

– Step 5: Generation of the private code

To ensure robust cryptographic strength and system-wide uniqueness, the private code p_c is generated using a nonlinear polynomial transformation of ϕ,

$$p_c = 3\phi^4 - 2\phi^2 + 7I \quad (4)$$

where, I is the identity matrix, that keeps the operations within the defined cryptographic space. The use of a nonlinear polynomial function in a finite matrix space ensures strong resistance to algebraic attacks while keeping the implementation efficient. It also guarantees that small changes in ϕ (e.g., due to different Λ or ν) produce large, unpredictable shifts in p_c enhancing security.

– Step 6: Encoding and smart contract commitment

The DP generates a private code p_c and transmits it to the smart contract. A smart contract receives private code and sums it up as p_c. The encoded private code is defined by,

$$p_{c_{enc}} = Alg_{enc}(p_c \times \phi \times \Lambda \times M_{id}) \quad (5)$$

the encoded private code $p_{c_{enc}}$ is embedded in the RB, linked to the dataset and access record. The DP then transfers the smart contract suite to the blockchain, enabling secure, transparent, and traceable execution of identity validation and data access permissions. Encoding with ϕ, Λ, and M_{id} ensures that only the original environment and user configuration can decode and verify the identity. This protects against replay attacks, cross-site forgery, and unauthorized mirroring.

3.2.3. Data preparation and encoding

Once identity verification is completed, the DP proceeds to prepare and encode the data for decentralized storage and controlled sharing. This stage is designed to achieve three primary goals: bind the data to its originating mirror site; ensure secure encryption and verifiability; Synchronize access and retrieval across multiple nodes. This step addresses the knowledge gap in traceable, mirror-specific data encoding and ensures that each data instance is verifiably linked to its origin while remaining secure during transmission and storage.

Step 1: Dataset encoding

To begin, the DP selects a term list t_l, derived from the dataset Δ, which defines both access control logic and searchable attributes. The data is also associated with, a data-specific mirror identifier m_d and a content encoding code τ. The encoded data is expressed as,

$$\Delta_{enc} = Alg_{enc}(\Delta \cdot t_l + m_d) + \tau \quad (6)$$

the use of a term list allows dynamic rule embedding and query optimization, while AES-based encoding ensures confidentiality. The additive term τ injects controlled randomness into the encryption output to support differential uniqueness per mirror.

– Step 7: Swarm upload and path encoding

After encryption, Δ_{enc} is uploaded to swarm. Upon successful storage, a **content path** Δ_{path} is returned to the DP. To bind this path securely to the system, the encoded data path is computed using,

$$\Delta_{path_{enc}} = Alg_{enc}(\Delta_{path} \cdot \tau + m_l) + \beta \quad (7)$$

where, m_l is the Path-specific mirror ID; β is a random matrix factor.

– Step 8: Synchronization and hash generation

The DP generates a hash synchronization H_s to ensure synchronization across mirror sites based on DC identifier Δ_{PID}, mirror site identifier m_{id}, and timestamp T_s. Synchronization hash is given by,

$$H_s = BLAKE2(\Delta_{PID} \cdot m_{id} \cdot T_s) \quad (8)$$

where, Δ_{PID} is customer identifier; T_s represents the timestamp. This hash ensures data cleanness and mirror-aware validity, resolving another key weakness of traditional PID-only systems.

– Step 9: Encoded content metadata

An intermediate content code N_{enc} is generated to bind metadata and identity information, which is denoted as,

$$N_{enc} = Alg_{enc}(F_{PID} + \beta \cdot m_k) + \tau \quad (9)$$

The framework persistent identifier F_{PID}, the random matrix factor β, and the mirror site key m_k are added together and encoded, along with the content encoded code, the content encoded code τ. This code is used in constructing the final data packet and adds a binding layer between framework identity and data logic.

– Step 10: Consensus validation across mirrors

To ensure all mirror nodes recognize the update consistently, a consensus check value V_c is computed as,

$$V_c = BLAKE2(H_s \cdot F_{PID}) \quad (10)$$

this hash-based validation ensures that all data changes are cryptographically traceable and agreed upon across nodes before being considered final.

– Step 11: Encoded data packet generation

The final encoded data packet e_{dp} is then constructed, expressed as,

$$e_{dp} = Alg_{enc}(\Delta_{path_{enc}} \cdot N_{enc} + m_m) + \rho \quad (11)$$

where, m_m is a metadata-specific mirror identifier and ρ a random code based on AES. The encoded code N_{enc}, encoded data path $\Delta_{path_{enc}}$, and this metadata-specific mirror identifier m_m are combined, and the encoding algorithm is applied to the combined data. An AES code ρ is aleatory engendered, added to the encoded data to generate the encoded data packet. This packet is what will ultimately be referenced in the smart contract for access and verification.

– Step 12: Encoded search identifier generation

To support mirror-specific query functions, the DP generates a search PID Z_{enc}, by combining the term list t_l, the random matrix factor β, and the mirror site identifier m_{id}, which is denoted as,

$$Z_{enc} = \theta \cdot Alg_{enc}(t_l \cdot \beta \cdot m_{id}) \quad (12)$$

the term list t_l, the random matrix factor β, and the mirror site identifier m_{id} are combined, and the encoding operation is applied to the result. The encoded data is combined with the factor θ in order to produce an encoded search persistent identifier. This search PID allows consistent data discovery and rule-matching in the BHRDS system.

3.2.4. Credential generation

The credential generation phase enables the DC to securely construct and submit a verifiable access request. This stage addresses a key requirement of the BHRDS framework: Binding access rights to cryptographically secure credentials, unique to both the user and the mirror site.

– Step 13: Decoding the private code

Upon receiving the encoded private code $p_{c_{enc}}$, the DC decodes it using the site-specific cryptographic key m_k and a secure decoding algorithm Alg_{dec}. The recovered private code p_c is expressed as,

$$p_c = Alg_{dec}(p_{c_{enc}} \cdot m_k) \quad (13)$$

the use of a site-bound key m_k ensures that only the intended mirror node can successfully decode and use the credential, reinforcing system-level security and preventing code reuse across mirrors.

– Step 14: Generating access credential

The DC produces an access credential τ by including the private code p_c, term list t_l, and destination mirror ID m_v, as input, and it is got as,

$$\tau = t_l + (p_c \cdot \beta \cdot m_v) \quad (14)$$

Additionally, DC produces the access credential based on t_l and invokes the smart contract for searching process. This credential encapsulates who is requesting access, where the access should occur (mirror site), and what rule set applies (via t_l). This composite credential guarantees that access requests are non-replayable, verifiable, and context-aware. Any mismatch in β, m_v, or p_c results in credential invalidation, ensuring access control integrity.

3.2.5. Operational management

The operational management stage governs the runtime interactions between the DC and DP, ensuring secure, rule-compliant execution of data access, upload, search, deletion, and withdrawal procedures through the smart contract system. This module guarantees that all activity within the BHRDS framework is authenticated, verifiable, and revocable, thereby maintaining the integrity and security of the system.

– Step 15: Customer registration and authentication

Upon receiving a registration request from a DC, the DP validates the provided credentials (as detailed in section 3.2.4). Once verified, the DP logs the customer's: PID, associated codes, and assigned profile vector, into a dedicated smart contract record. This on-chain registration ensures that only authorized users can interact with the system. If a customer fails authentication or becomes inactive, the DP can revoke access by removing their record from the authorization set. On-chain registration with tamper-proof logs allows traceable user access control, eliminating the risk of unauthorized mirror or external access, which is a critical limitation in traditional decentralized storage systems.

– Step 16: Content upload and search identifier generation

During content upload, the DP selects a relevant term list t_l to define access or search rules and generates a search persistent identifier. This identifier is composed of an encoding code (from section 3.2.3), the operation PID, and an encoded search PID. All three components are stored within a smart contract to enable efficient content retrieval, rule enforcement, and traceability.

– Step 17: Secure deletion and search handling

The DP can initiate content deletion by referencing: the operation PID, and the associated search persistent identifier. This enables bulk deletion of all content tied to a particular rule set or outdated identifier. Such fine-grained control helps remove obsolete or irrelevant data, freeing storage and preventing unnecessary exposure. For search operations, the system accepts an encoded search PID as input. The smart contract decodes this and returns: the relevant operation PID, and the content pointer, thus enabling secure and targeted retrieval. This granular search-deletion linkage is vital for maintaining long-term compliance, as datasets evolve or become deprecated over time.

– Step 18: Withdraw function for search service

The withdraw function allows the DP to disable an existing search service tied to a dataset or term list. This revocation is useful when: datasets are no longer shareable due to policy updates, contract's expiry, or data owners request de-listing. Withdrawals are handled on-chain and logged for audit purposes. This feature adds dynamic governance and lifecycle management to the system, which is typically missing in rigid smart contract deployments.

3.2.6. Data integrity verification

The data integrity verification phase ensures that all operations carried out by the DC are authorized, verifiable, and synchronized across mirror sites. It confirms that credentials have not been forged or reused, and that all transactions are timestamped and uniquely traceable. This step addresses the crucial need for cross-mirror consistency and fine-grained access auditing, identified earlier as gaps in decentralized data sharing systems.

– Step 19: Credential verification

During operation execution, the smart contract receives the access credential generated by the DC and compares it with a locally computed reference credential τ^*, stored during contract creation, which is specified as,

$$\tau^* = t_l + \left(p_{c_2} \cdot \beta \cdot m_v\right) \tag{15}$$

the access credential τ^* is saved with a smart contract. It is generated similarly to the access credential τ using a different private code p_{c_2} but the same term list t_l, the random matrix factor β, and the access credential-specific mirror identifier m_v. The smart contract contains all the deals. It is used to check the access credential that DC authorized. The smart contract grants access if and only if $\tau = \tau^*$, DC can operate. By reconstructing τ^* internally, the system avoids exposing secrets while enabling deterministic credential validation. This method ensures access is bound to the correct profile, site, and session context.

– Step 20: Timestamp encoding

At the same time, the timestamp T_s is encoded to guarantee the veracity and consistency of processes across mirror sites. The encoded timestamp is expressed as,

$$T_{enc} = Alg_{enc}(T_s \cdot m_v) \tag{16}$$

this encoded timestamp is used in logs to verify synchronization across decentralized ledgers and detect tampering or replay attempts.

– Step 21: Encoded operation and credential validation

Once access is approved, the DC submits an encoded request combining both the operation and credential values. This encoded token Q consists of two parts, is expressed as,

$$Q = Alg_{enc}(O_{PID} \cdot \tau \cdot m_x) + Alg_{enc}(\rho \cdot Z_{enc} \cdot m_x) \quad (17)$$

where m_x represent an identifier for an operation on a specific mirror; O_{PID} is the operation persistent identifier; τ is the access credential; and m_x is transaction-specific mirror identifier. They are combined and processed using the encoding algorithm Alg_{enc}. Besides, the AES code selected arbitrarily ρ, is combined with the encoded search persistent identifier Z_{enc} and transaction-specific mirror identifier m_x, and the result is also encoded with encoding algorithm. The encoded data sets are then combined, and the resultant value represents the encoded operation and access credential Q. The first part binds the operation to the credential and target node, while the second ensures that the search was conducted securely using a verifiable identity and encrypted path. Combining operation logic, credentials, and mirror-specific identifiers within a double-encoded structure prevents replay attacks, credential reuse, or unauthorized command injection—a critical upgrade over basic smart contract-based access system

– Step 22: Result transmission

After validation, the smart contract computes the result (e.g., content pointer or permission flag) and securely returns it to the DC. All transactions are logged for auditability.

3.2.7. Authorization and data recovery

The Authorization and Data Recovery stage is the final verification layer in the BHRDS framework. It ensures that the DC, after completing credential generation and access validation, is fully authorized to retrieve and use the requested data. This mechanism provides cryptographic confirmation of identity, transaction intent, and agreement between parties before any data is accessed or recorded.

This step addresses the core requirement for end-to-end trust and verifiable authorization in decentralized systems-closing the loop from identity binding (3.2.2) to access enforcement (3.2.6).

– Step 23: Agreement condition computation by DC

To initiate the agreement process, the DC generates an agreement condition A_{cond}, which acts as a unique fingerprint of the authorization session. It denoted as,

$$A_{cond} = Q + BLAKE2\left(\Delta_{PID} \cdot (\nu \times p_c \times m_v)\right) \quad (18)$$

where m_v is a validation-specific mirror identifier, is combined with the random integer ν, and the private code p_c, also combined with the DC identifier Δ_{PID}. The resulting value is hashed using the BLAKE2 operation. Then added to the encoded operation and credential value Q. This composite hash ensures that authorization is session-specific, non-replayable, and uniquely tied to the DC's verified credentials. The use of BLAKE2 adds cryptographic resistance to tampering and impersonation.

– Step 24: Smart contract agreement matching

Once the DC submits A_{cond}, the DP (or the smart contract on behalf of the DP) generates its own reference agreement condition $A_{cond'}$ using internal records,

$$A_{cond'} = Q + BLAKE2\left(\Delta_{PID'} \cdot (\nu \times p_{c_2} \times m_v)\right) \quad (19)$$

this agreement condition in the smart contract $A_{cond'}$ is generated similarly to the agreement condition A_{Cond}, using a different DC identifier $\Delta_{PID'}$, private code p_{c_2}, but the same random integer ν and validation-specific mirror identifier m_v. Recomputing the agreement condition within the smart contract ensures that all validation logic remains internal, decentralized, and verifiable without requiring manual off-chain checks.

– Step 25: Final validation and data recovery

If the two agreement conditions match: $A_{cond} = A_{cond'}$, Then the customer is verified. The smart contract grants access to the requested dataset, the data pointer from swarm is retrieved, and the DC is registered as an authorized user in the blockchain ledger for audit logging. This final check ensures that only authenticated users who have completed all prior cryptographic steps can recover data. It also adds a non-repudiable audit trail for regulatory compliance.

3.2.8. Deal finalization

In the final stage of the BHRDS workflow, the DC receives and decrypts the verified data package following a completed authorization agreement. This ensures that only an authenticated DC who has passed all previous cryptographic and contractual validations is able to retrieve and reconstruct the original dataset.

– Step 26: Encoded validation confirmation

The DP transfers the agreed operation token Q and agreement condition A_{cond} to the DC. The DC confirms receipt by computing a final encoded validation factor $A_{cond_{enc}}$,

$$A_{cond_{enc}} = Q_{ac} + BLAKES2\left(\Delta_{PID'} \cdot (v \times p_c \times m_v)\right) \quad (20)$$

the private code p_c, random integer v and the validation specific mirror identifier m_v are combined, and the resultant value is hashed with another DC identifier $\Delta_{PID'}$ using the BLAKE2 hash function. Finally, the result is combined with another encoded operation and access credential Q_{ac}. This validation check confirms that the entity requesting decryption is the same one previously authenticated via smart contract, mitigating any risk of response hijacking or impersonation. If the computed $\mathrm{A_{cond_{enc}}}$ matches the expected result on DP's side, the DP proceeds with data delivery.

– Step 27: Transfer of encoded data and decryption keys

The DP checks the valid content provided by the DC. In case is identic to the content recorded in the DP. The DP transfers the encoded data path and encoded code to the DC. The DP combines the encoded data path and encoded code and transfers them to swarm along with K_s. These components are transmitted securely and registered on-chain for traceability. The DC then retrieves the encrypted dataset from swarm using the encoded data path.

– Step 28: Data recovery and decoding

Once the encrypted dataset is downloaded from swarm, the DC applies the decoding function to extract the original dataset. The content is reconstructed using the decoding algorithm Alg_{dec}, based on the received components,

$$\Delta = \mathrm{Alg_{dec}}(\Delta_{\mathrm{recovered}} + \tau \cdot \mathrm{m_d}) \quad (21)$$

the decoded data Δ is obtained by performing a decoding operation on the combination of the recovering encoded data $\Delta_{recoverd}$, the content encoded code τ, and the data-specific mirror identifier m_d. The content encoded code τ, the recovered data $\Delta_{recovered}$ and the data specific mirror identifier m_d are combined to acquire the data file.

$$\Delta = \Delta_{recovered} \cdot \tau \cdot m_d \quad (22)$$

At last, data content is recovered back through DC. Binding the decoding process to τ and m_d ensures that only the right recipient on the correct mirror node can reconstruct the file, adding final-layer cryptographic assurance to the data sharing process. Once decoded, the original dataset Δ is recovered, concluding the secure transaction between DP and DC. The steps of the proposed method are illustrated by the Algorithm 1.

Algorithm 1– BHRDS workflow

Input: Dataset Δ, Term list t_l, Customer ID, Mirror IDs, Key
Output: Authorized, verifiable access and recovery of data
1 Initialize system
    Generate F_PID and Root PID Ξ
    Store encoded Ξ on blockchain
2 Register customers
    Issue PID, access code, and profile vector Λ
    Compute private code p_c and store securely
3: Prepare data
    Encrypt Δ, combine with t_l and mirror ID
    Upload to swarm and register metadata on-chain
4: Generate credential
    DC computes τ using p_c and session data
    Submit τ to smart contract

5: Validate access
Smart contract verifies τ and agreement condition
If valid → grant access to data
6: Recover data
Retrieve from swarm
Decode using τ and mirror ID
Output original dataset Δ

4. RESULTS AND DISCUSSION

In this section, we conduct extensive experiments to demonstrate the validity of our proposed model. Moreover, the experimental setup, performance evaluation, as well as discussion are developed in this section.

4.1. Experimental environment

This section analyzes the performance of the proposed BHRDS framework. The development of the system is performed using the PYTHON tool within Windows 11 OS, 32 GB RAM, and Intel Core i7. The algorithms and cryptographic functions were developed with NumPy, PyCryptodome, and custom matrix-based encoding modules. BLAKE2 cryptographic hash function was used due to its high speed and security. It supports keying, salting, personalization, and hash tree modes and can output digests from 1 up to 64 bytes. Three distinct key lengths of advanced encryption standard (AES) were utilized to assess the system's performance under varying cryptographic loads, 128, 192, and 256 bits. The longer the key, the more secure it is, but the slower the encryption and decryption will be as well. Knowing the impact of encryption on latency and validation time is vital. They were implemented using the PyCryptodome, a self-contained Python package of low-level cryptographic primitives. The handle server version 9.3.1 was distributed across various mirror nodes using Docker containers for the handle system working environment, with the PyHandle Python client managing persistent identifier operations (create, read, update, delete). The RB was mimicked using a private Ethereum testnet with smart contracts generated via Remix IDE and connected with Web3.py. It allowed Python scripts to deploy and interact with smart contracts (e.g., for identity, credential, and access management). Each smart contract handled registration, identity binding, access restriction, and logging. The decentralized storage layer was deployed using swarm, where encrypted data packets were uploaded, and metadata pointers were saved on-chain. An HTTP API endpoint was used to upload and retrieve the data. Python scripts are used to process them. The components use Jason's message to transmit identifiers and metadata. Furthermore, a smart contract referenced the swarm routes and handle PIDs to provide consistency and verifiability. A Docker is used to containerize the handle server, the Ethereum, and the decentralized storage network, swarm. This procedure ensures a consistent and effective deployment.

In the simulation scenario, a DP registers mirror sites within the handle system and gets the persistent identifiers. After applying for access, the DC will receive the profile credential. The DP encrypts datasets based on mirror-specific parameters and uploads them to swarm. The blockchain stores hash, encoded identifiers, access conditions, and smart contracts to enforce smart contracts. The number of blocks in the Blockchain varies in two ranges: 50 to 200 and 300 to 1,000. We will use two instances to analyze the rising number of consumers (NC). It is associated with the fluctuation of the number of blocks. One is rising from 10 to 100 users, and the other is from 500 to 10,000 people. User generation was achieved using Python scripts with the Faker package to construct synthetic identities and Web3.py to register them on the blockchain smart contract. The performance metrics included irregularity detection rate (ratio of the number of the anomalies detected to the total number of anomalies present in the data in percentage) and response time (time taken to react to a given request). Docker-based isolation ensured reproducibility, while simulated attack vectors and replay attempts tested the security resilience of the access protocols. This arrangement enables a complete examination of BHRDS in terms of scalability, cryptographic complexity, and secure interoperability.

4.2. Result interpretation

The performance of the developed technique is analyzed based on the proportions of irregularity detection and also the time takes to complete a request. The irregularity detection proportion is referred to as the total number of customers (NCs) identified as fake regarding the whole NCs. The response time is the promptness with the RB and swarm chain, which executes requests during a given time. The number of blocks in the Blockchain is varied in two ranges: from 50 to 200, and from 300 to 1,000. We will consider two cases to analyze the increasing NCs. It is along with the variation of the number of blocks. One is creasing from 10 to 100 users and the other is from 500 to 10,000 users. Blockchain length (BCL) is the total number of all the blocks forming the chain. The length of blocks is an important aspect of the whole chain

because it influences all the operations in the network. The key lengths for the AES algorithm are set to 128, 192, and 256 bits. These variations are conducted to analyze the effectiveness of the developed technique under different scenarios and ensure its robustness and efficiency in various practical situations.

4.2.1. Number of blocks between 50-200 and customers number between 10 to 100

In the first case, the BCL is 50 to 200, and the user's number increases from 10 to 100. Table 1 indicates that irregularity detection remained consistently strong across all encryption scenarios. As the number of users increased, detection decreased slightly but remained high. Most notably, detection improved with stronger keys: at 100 users and 50 blocks, detection increased from 90.44% (AES-128) to 92.67% (AES-256). This trend was consistent across all block sizes. The improvement is linked to the framework's use of BLAKE2 hashing and structured profile vectors, which increases collision resistance and irregularity sensitivity. Encoding mirror-specific identifiers and identity parameters are designed to add a unique, non-reversible trace to each transaction, significantly boosting the anomaly detection capability.

Meanwhile, Table 2 shows that response times were consistently lower with higher key strengths. At 100 users and 200 blocks, response time dropped from 31.62 seconds (AES-128) to 29.00 seconds (AES-256). This efficiency is due to the design, where the validation based on smart contract decouples computation from consensus, allowing for off-chain resolution with minimal on-chain latency.

Table 1. Irregularity detection rate (%) for BCL 50 to 200 and customers 10 to 100

Number of blocks	key lengths	Customers number			
		10	30	60	100
50	128	94.21	93.46	92.40	90.44
	192	95.52	95.22	93.88	91.94
	256	96.71	95.95	95.37	92.67
100	128	93.38	92.88	91.53	89.24
	192	94.72	94.07	92.48	91.28
	256	96.47	95.27	93.82	91.88
150	128	92.88	91.58	90.06	88.70
	192	94.36	93.45	91.86	89.87
	256	94.98	94.64	93.19	90.93
200	128	91.42	90.93	89.49	87.48
	192	92.84	92.31	91.06	89.07
	256	94.32	93.91	91.82	90.66

Table 2. Irregularity detection rate (%) for BCL 50 to 200 and customers 10 to 100

Number of blocks	Key lengths	Customers number			
		10	30	60	100
50	128	94.21	93.46	92.40	90.44
	192	95.52	95.22	93.88	91.94
	256	96.71	95.95	95.37	92.67
100	128	93.38	92.88	91.53	89.24
	192	94.72	94.07	92.48	91.28
	256	96.47	95.27	93.82	91.88
150	128	92.88	91.58	90.06	88.70
	192	94.36	93.45	91.86	89.87
	256	94.98	94.64	93.19	90.93
200	128	91.42	90.93	89.49	87.48
	192	92.84	92.31	91.06	89.07
	256	94.32	93.91	91.82	90.66

4.2.2. Number of blocks between 50-200 and customers number among 500 to 10,000

In the second case, involving 500 to 10,000 users with the same block range (50 to 200), Figure 4 highlights that even at maximum user load. For instance, in Figures 4(a)-4(d), the system maintains high detection: 80.95% with AES-256, compared to 75.22% with AES-128. The graceful degradation in detection rates indicates strong stability in BHRDS's decentralized validation layer, where encoded identifiers resist spoofing even under stress.

Simultaneously, Figure 5 demonstrates that response times remain within operational standards, although higher than in low-load conditions. In Figures 5(a)-5(d) with 10,000 users and 50 blocks, AES-256 completes in 41.24s, showing that the modular nature of swarm off-chain storage effectively decouples data retrieval from blockchain congestion.

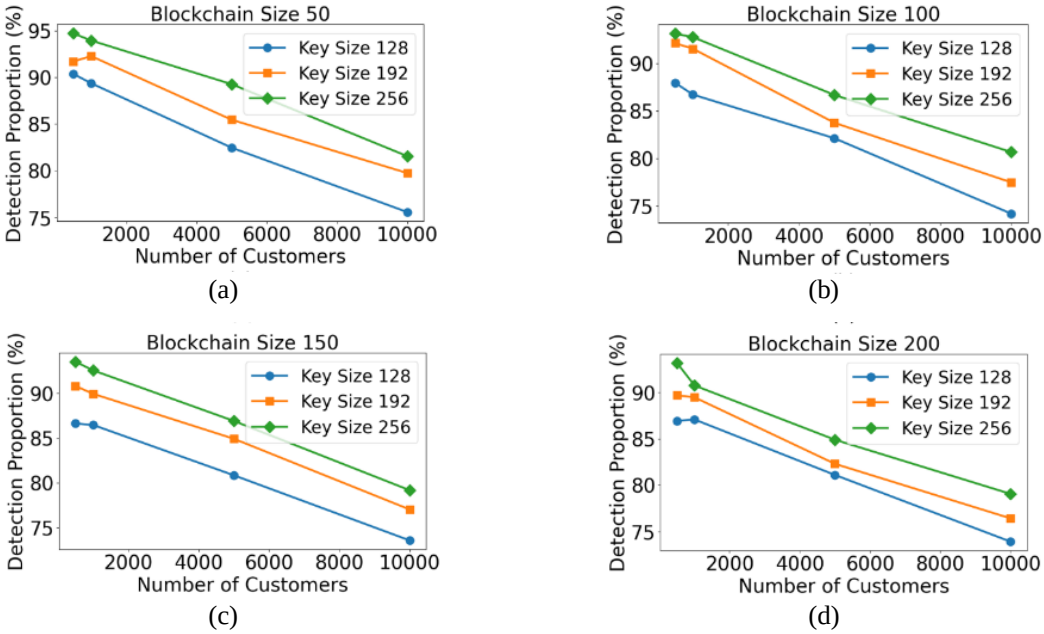

Figure 4. Irregularity detection rate through 500 to 10,000 customers within different blocks size, (a) 50 blocks, (b) 100 blocks, (c) 150 blocks, and (d) 200 blocks

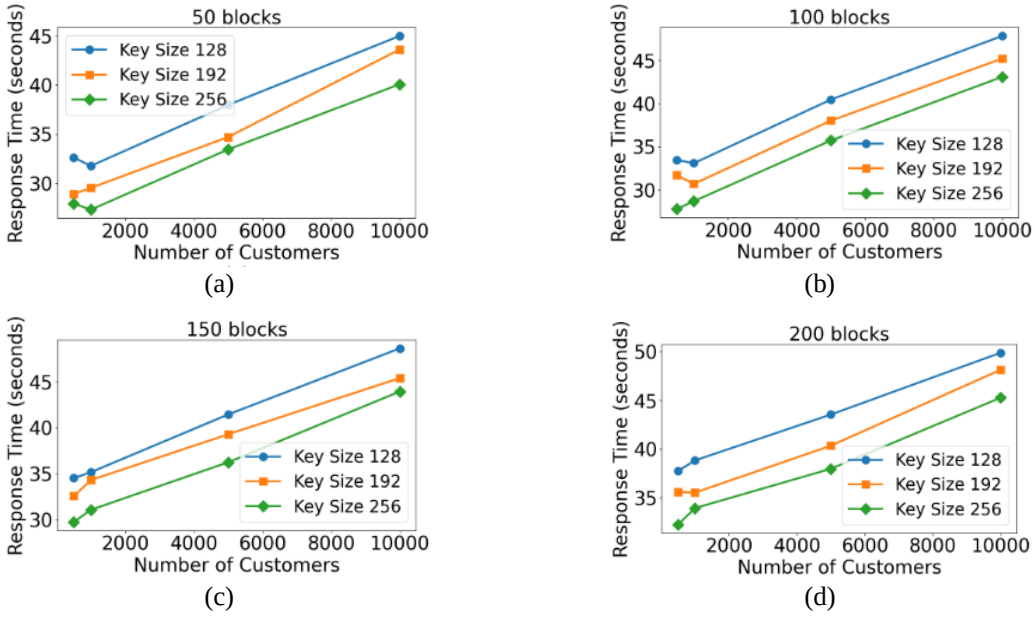

Figure 5. Response time customers number amid 500 to 10,000 under varying block size, (a) 50 blocks, (b) 100 blocks, (c) 150 blocks, and (d) 200 blocks

4.2.3. Number of blocks between 300-1,000, number of customers 10 to 100

The third scenario, where a longer blockchain (300-1,000) and a smaller user population (10-100), is illustrated in Figure 6. With longer chains, detection rates were high, at least 82% at capacity, meaning BHRDS's mirror-sensitive cryptographic identifiers are stable even as blockchain depth increases, as shown in Figures 6(a)-6(d). These results support the hypothesis that longer chains, while adding latency, do not compromise the system's ability to detect irregularities when identity and data paths are tightly bound.

Corresponding response times shown in Figure 7 also reflect manageable performance costs. Even with 100 users and 1,000 blocks, the response remained under 36.99 seconds, validating the role of swarm and the lightweight PID resolution strategy in mitigating the typical load issues associated with long blockchains, as shown in Figures 7(a)-7(d).

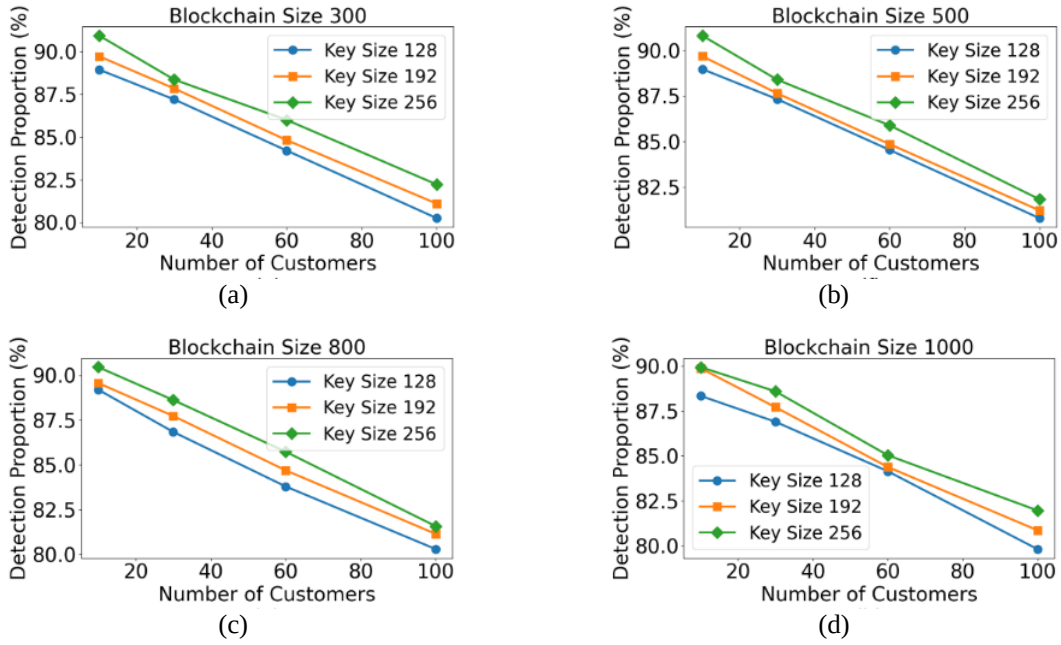

Figure 6. Irregularity detection rate with customers balancing between 10 and 100 customers within, (a) 300 blocks, (b) 500 blocks, (c) 800 blocks, and (d) 1,000 blocks

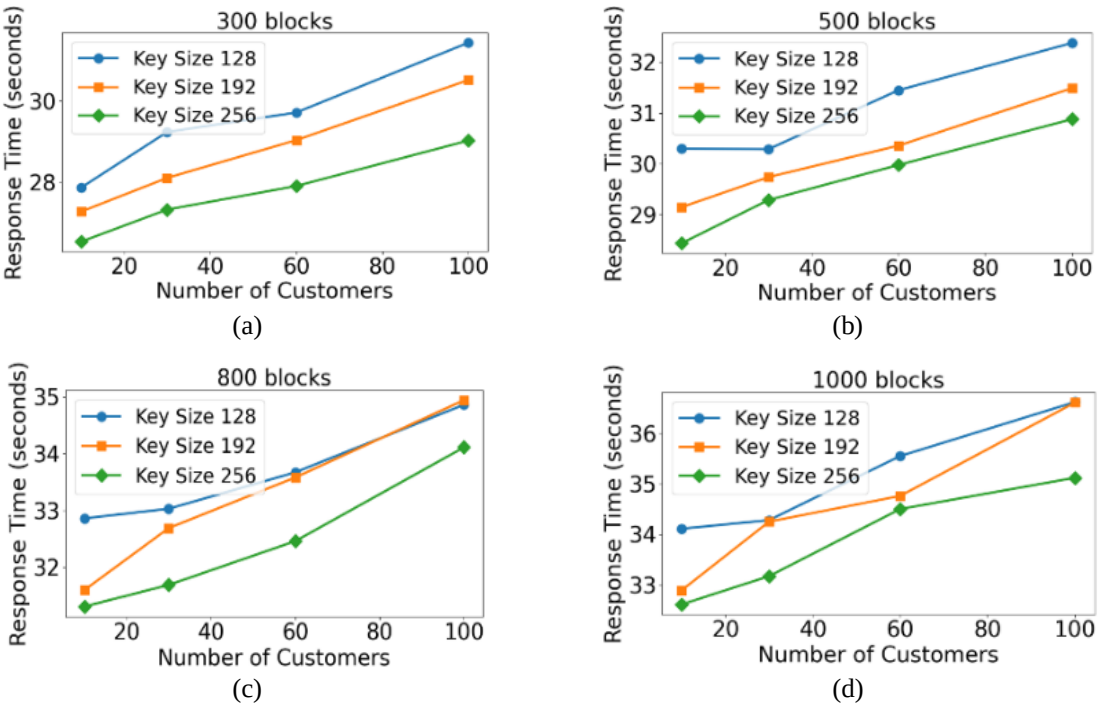

Figure 7. Response time with vary between customers 10 to 100 through diverse block size, (a) 300 blocks, (b) 500 blocks, (c) 800 blocks, and (d) 1,000 blocks

4.2.4. Number of blocks between 300-1,000, number of customers 500 to 10,000

Finally, the fourth scenario combined the highest loads on both dimensions: 500 to 10,000 users and BCLs of 300 to 1,000 as shown in Figure 8. Figures 8(a)-8(d) shows that, even under these extreme conditions, detection rates remain above 77% with AES-256. BHRDS does not fail or become unstable, reinforcing its capacity for real-world, high-throughput deployments.

Regarding response, Figure 9 shows that at 10,000 users and 1,000 blocks, the system achieves a response time of 55.83s with AES-128, reduced to 53.51s with AES-256. This minimal increase in latency-relative to the complexity of the environment-confirms the value of the proposed architecture, particularly its ability to localize decision-making and verification within mirror nodes (Figures 9(a)-9(d)).

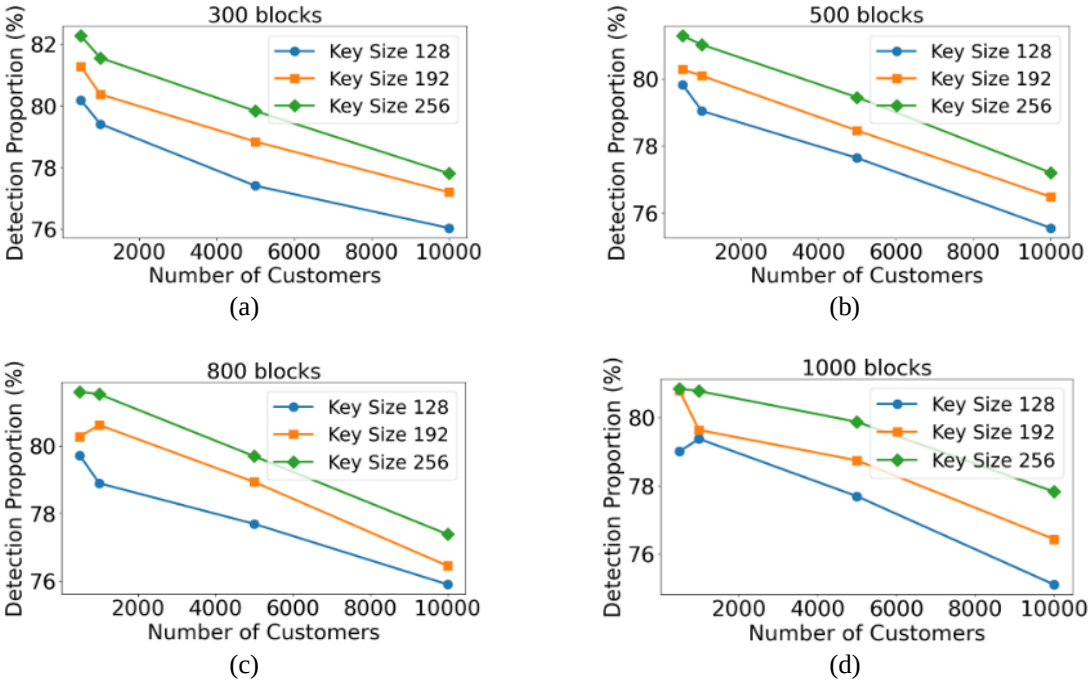

Figure 8. Irregularity detection rate with BCL, (a) 300 blocks, (b) 500 blocks, (c) 800 blocks, and (d) 1,000 blocks, and customers 500 to 10,000

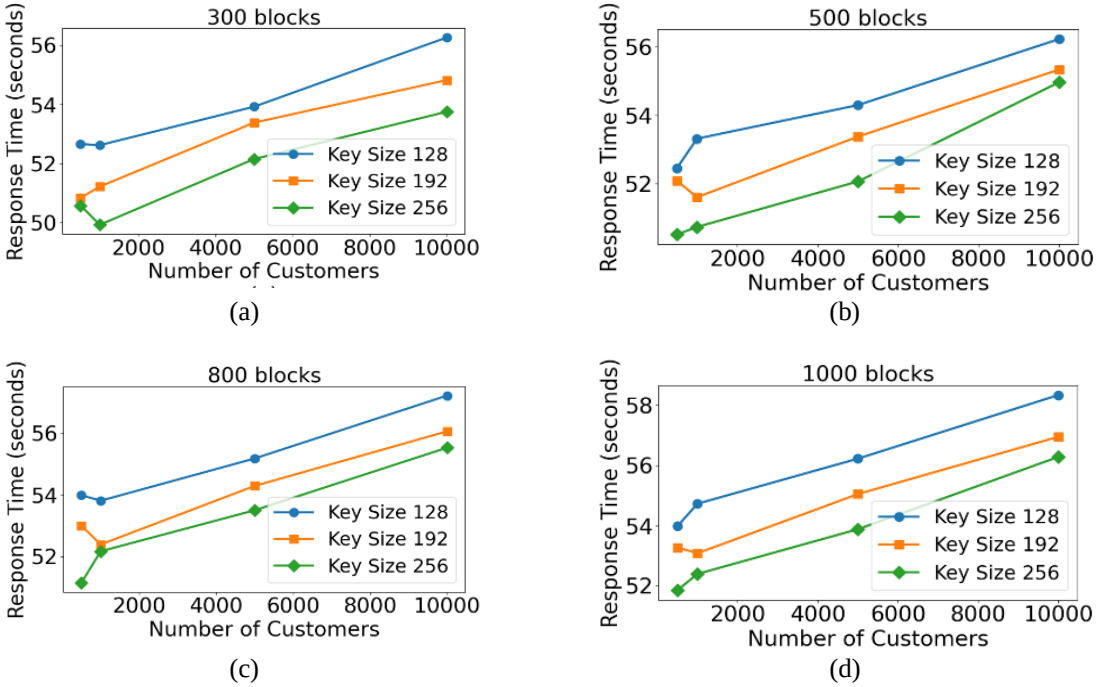

Figure 9. Response time with BCL setup at: (a) 300 blocks, (b) 500 blocks, (c) 800 blocks, and (d) 1,000 blocks; and customers increasing from 500 to 10,000

These experimental results demonstrate that BHRDS performs consistently across various operational settings. Not only does the system scale well with user numbers and BCL, but it also benefits from stronger encryption, improving detection rates and processing efficiency. These results validate the framework's core design principles, a well-justified, FAIR-compliant solution and support its suitability for real-world RD -sharing environments where scalability, privacy, and traceability are essential.

4.3. Comparing BHRDS with the existing similar model

In this subsection, we present the results of a comprehensive experimental study comparing our proposed system, with other existing models, MedRec [13] (medical data management on the blockchain), HSHB [20] (Hybrid blockchain-based health data sharing method), SPDS [15] (secure and auditable private data sharing), and IoVChain [17] (blockchain-based internet of vehicles data secure sharing scheme). These models were evaluated across four critical performance metrics, the average throughput, the average latency, the fault tolerance, and the total security breaches.

A comparative analysis was conducted through experiments. Each system was deployed by means of virtual machines (VMs) to form a networked system of nodes. These VMs were then connected via a virtual network with performance controls for latency and bandwidth, to coat the replicated environment of scattered different locations. We simulated the network latency to vary between 10 milliseconds and 100 milliseconds to test under different levels of network congestion. A custom transaction generator was developed to produce a continuous stream of transactions. We initially subjected each system to 1,000 transactions per second (TPS) and then continued to scale the load until we reached a point where each system could no longer handle any additional connections. All systems used the same block size, which were blocks of 10 transactions. To maintain realism about block generation activity reflecting that of actual blockchain operations, the creation intervals for blocks were specific to each system's consensus algorithms.

The experimental results of the BHRDS framework reflect substantial improvements over existing data-sharing models, particularly in terms of detection accuracy, scalability, and decentralized access control. In comparison to the other encryption-based systems and blockchain-enabled models, BHRDS offers a more robust and flexible architecture by integrating persistent identifiers (PIDs), smart contracts, and mirror-specific encryption strategies. Compared to the traditional handle system, which require centralized authorities, BHRDS eliminates this dependency by using matrix-based profile vectors and dynamic credential encoding. This approach enables independent identity management per mirror site without relying on centralized trust anchors, making the system more adaptable to federated and cross-institutional environments. Frameworks such as MedRec [13] and SPDS [15] introduced blockchain for access auditing and transparency, but their designs were typically tailored for specific domains (e.g., smart grid or healthcare) and lacked PID integration or mirror-level security differentiation. BHRDS extends this paradigm by coupling the handle system with smart contracts, allowing identifiers to be securely assigned, verified, and revoked across multiple distributed nodes. The throughput is defined as the transactions treated in a second. The latency is the time from submission of a transaction to its inclusion in the block confirmation. The fault tolerance algorithms simulate the random failures of nodes and check how many can continue to participate. The security breaches analyze the attacks, which happen randomly (for example 5% of chance) during transaction processing. Figure 10 shows that BHRDS outperforms SPDS, TEBDS, and MedRec in irregularity detection and credential validation latency. For instance, BHRDS has the high throughput (180 tps) and low latency (0.015 s), outperforming all systems. The SPDS is following while MedRec, and IoVChain lagged behind due to its design choices, which are inherently not scalable. The analysis shows that BHRDS has maximum fault tolerance (95%) and remains operational during node crashes. Furthermore, SPDS has robustness, while IoVChain and HSHB show weaknesses. It is evident that the network design and consensus mechanisms affect fault tolerance. BHRDS came in with the lowest number of security breaches (3), indicating it has strong measures. This is largely due to the lightweight cryptographic operations and local validation logic encoded in smart contracts, which reduce dependency on network consensus delays for access decisions. Moreover, while existing systems focus on static identities or token-based access, BHRDS introduces mirror-specific encryption and operation-specific credentialing, which significantly reduce the risk of unauthorized reuse, replay, or impersonation—especially important in multi-party research environments.

4.4. Practical implication of the model

The results of this study confirm the effectiveness of BHRDS as a decentralized, secure, and scalable data-sharing framework capable of operating across distributed research infrastructures. The system supports fine-grained access governance by integrating persistent identifiers (PIDs) from the handle system with smart contract-based access control and cryptographically encoded identity profiles while maintaining transparency and auditability. One of the key implications of this work is its potential alignment with the FAIR data principles-particularly in enhancing findability and accessibility through persistent, verifiable

identifiers and enabling reusability via transparent, rule-based access conditions. These capabilities position BHRDS as a foundational component for future federated research platforms, cross-institutional collaborations, and IoT-driven data ecosystems, where data security, traceability, and trust are essential. Furthermore, the modularity of BHRDS allows it to be adapted beyond the academic context, such as in healthcare, smart grid, and governmental registries, where policy-driven data access must be enforced transparently and verifiably. Furthermore, the modularity of BHRDS allows it to be adapted beyond the academic context, such as in healthcare, smart grid, and governmental registries, where policy-driven data access must be enforced transparently and verifiably.

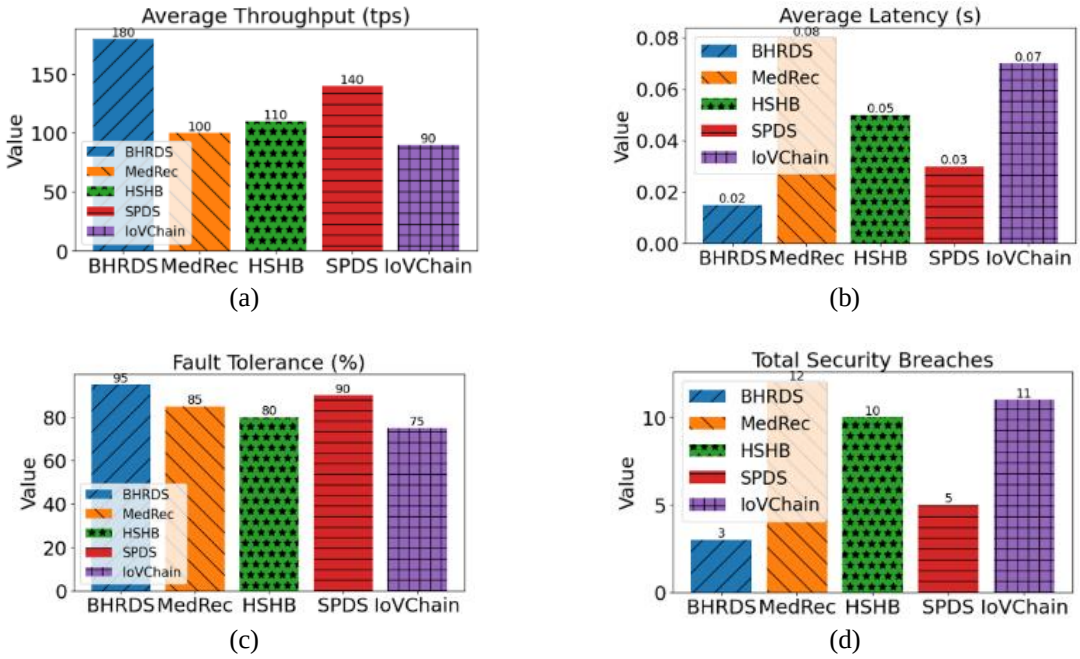

Figure 10. Comparison of the, (a) average throughput, (b) average latency, (c) fault tolerance, and (d) total security breaches of BHRDS with existing similar models

4.5. Limitation and potential research area

While BHRDS demonstrates considerable advantages, it also shares several limitations common to blockchain-based frameworks. One notable constraint is the potential performance bottleneck under single-chain conditions, especially when processing a high volume of credential submissions or concurrent data requests. In such environments, transaction throughput and confirmation latency could become critical concerns. Additionally, while the use of swarm for off-chain storage alleviates blockchain bloat, there may still be increased operational overhead when synchronizing metadata and access credentials across numerous mirror sites. As a result of this act, real-time applications could consume more resources and have a higher latency.

Another limitation lies in the complexity of integration with legacy data infrastructures and existing PID services, mainly when institutions use different identifier protocols or metadata schemas. Aligning BHRDS with global PID registries such as DataCite or ORCID may require custom adapters or policy negotiation layers. Furthermore, while the current system performs well in controlled test environments, it has not yet been deployed in a large-scale real-world scenario. Such a deployment would introduce variables like heterogeneous network conditions, adversarial behavior, and dynamic trust relationships, all of which warrant further study.

Despite these limitations, BHRDS is built with a modular and extensible architecture, which supports future integration with multichain and sharding environments. This adaptability, combined with its decentralized identity verification and encoded access logic, makes it a promising foundation for building more scalable, interoperable, and FAIR-compliant data sharing systems in the future. Upcoming research directions include exploring multichain architectures to distribute verification workloads and reduce processing delays and developing federated trust management mechanisms to allow different institutions to coordinate credential validation without relying on centralized authorities. Additionally, integrating

automated compliance auditing within smart contracts could help ensure access policies remain aligned with institutional or legal standards over time. Another critical research opportunity is improving real-time interoperability with existing RD repositories and PID services. Building adapters or bridges to globally recognized platforms (e.g., DataCite, ORCID, and CrossRef) could enhance adoption and create seamless, secure RD ecosystems. By addressing these future challenges, BHRDS has the potential to evolve into a robust backbone for next-generation trusted data-sharing platforms across sectors and disciplines.

5. CONCLUSION

This research introduced the BHRDS framework, a decentralized, blockchain-based approach to secure RD sharing, integrating persistent identifiers (PIDs) and smart contract-enforced access control. Through rigorous experimentation under varying user scales, encryption levels, and blockchain loads, BHRDS demonstrated consistent performance in irregularity detection and response time. These results validate the framework's scalability, cryptographic resilience, and ability to maintain access integrity across distributed mirror sites. Beyond performance metrics, the findings highlight BHRDS's broader potential in addressing the longstanding challenges of traceability, decentralized access governance, and FAIR-compliant data stewardship. By combining the handle system with programmable smart contracts and mirror-specific encryption, BHRDS offers a modular solution that bridges existing identifier infrastructures with the transparency and auditability of blockchain. For the research community, this work presents a pathway toward trusted, federated data-sharing environments where persistent access credentials can be verified independently, yet in a coordinated and secure manner. BHRDS can be extended to support collaborative scientific platforms, health data registries, IoT networks, and other domains where decentralized access control and verifiable identity binding are essential.

Future work will address key limitations such as reliance on a single blockchain layer by exploring multichain architectures, cross-ledger interoperability, and real-time trust negotiation among institutions. Furthermore, the integration of automated policy compliance and interoperability with global PID services like ORCID, DataCite, and CrossRef could significantly enhance adoption. BHRDS offers the foundation for a next-generation secure data-sharing architecture that aligns with open science ideals while ensuring traceability, integrity, and controlled access across scattered digital ecosystems.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Mahamat Ali Hisseine	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Deji Chen		✓				✓		✓	✓	✓	✓	✓		
Yang Xiao	✓		✓	✓			✓			✓	✓		✓	✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, Deji Chen, by request.

REFERENCES

[1] T. Wen, "Data Sharing," in *Encyclopedia of Big Data*, L. A. Schintler and C. L. McNeely, Eds., Cham: Springer International Publishing, 2022, pp. 335–338. doi: 10.1007/978-3-319-32010-6_322.

[2] M. D. Wilkinson *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," *Sci Data*, vol. 3, no. 1, p. 160018, 2016, doi: 10.1038/sdata.2016.18.

[3] M. A. Hisseine, D. Chen, Y. Xiao, and P. K. Alimo, "A review of digital object architecture and handle system: Development, current applications and prospective," *Internet of Things*, vol. 26, p. 101230, 2024, doi: https://doi.org/10.1016/j.iot.2024.101230.

[4] F. Liu and D. Panagiotakos, "Real-world data: a brief review of the methods, applications, challenges and opportunities," *BMC Med Res Methodol*, vol. 22, no. 1, p. 287, 2022, doi: 10.1186/s12874-022-01768-6.

[5] J. Wang *et al.*, "Data sharing in energy systems," *Advances in Applied Energy*, vol. 10, p. 100132, 2023, doi: https://doi.org/10.1016/j.adapen.2023.100132.

[6] G. Wang, Q. Liu, J. Wu, and M. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *Comput Secur*, vol. 30, no. 5, pp. 320–331, 2011, doi: https://doi.org/10.1016/j.cose.2011.05.006.

[7] H. Deng, Z. Qin, Q. Wu, Z. Guan, and Y. Zhou, "Flexible attribute-based proxy re-encryption for efficient data sharing," *Inf Sci (N Y)*, vol. 511, pp. 94–113, 2020, doi: https://doi.org/10.1016/j.ins.2019.09.052.

[8] T. Lakum and B. T. Rao, "Mutual query data sharing protocol for public key encryption through chosen-ciphertext attack in cloud environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, pp. 853–858, Feb. 2022, doi: 10.11591/ijece.v12i1.pp853-858.

[9] S. Handore, P. Kolapkar, P. Chavan, and P. Chavan, "Cost-effective anonymous data sharing with forward security using improved authentication," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 129–136, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp129-136.

[10] T. Neubauer and J. Heurix, "A methodology for the pseudonymization of medical data," *Int J Med Inform*, vol. 80, no. 3, pp. 190–204, 2011, doi: https://doi.org/10.1016/j.ijmedinf.2010.10.016.

[11] J.-J. Yang, J.-Q. Li, and Y. Niu, "A hybrid solution for privacy preserving medical data sharing in the cloud environment," *Future Generation Computer Systems*, vol. 43–44, pp. 74–86, 2015, doi: https://doi.org/10.1016/j.future.2014.06.004.

[12] B. K. Beaulieu-Jones *et al.*, "Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing," *Circ Cardiovasc Qual Outcomes*, vol. 12, no. 7, p. e005122, Jul. 2019, doi: 10.1161/CIRCOUTCOMES.118.005122.

[13] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using Blockchain for Medical Data Access and Permission Management," in *2016 2nd International Conference on Open and Big Data (OBD)*, 2016, pp. 25–30. doi: 10.1109/OBD.2016.11.

[14] Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, "BBDS: Blockchain-Based Data Sharing for Electronic Medical Records in Cloud Environments," *Information*, vol. 8, no. 2, 2017, doi: 10.3390/info8020044.

[15] Y. Wang *et al.*, "SPDS: A Secure and Auditable Private Data Sharing Scheme for Smart Grid Based on Blockchain," *IEEE Trans Industr Inform*, vol. 17, no. 11, pp. 7688–7699, 2021, doi: 10.1109/TII.2020.3040171.

[16] L. Yin, J. Feng, H. Xun, Z. Sun, and X. Cheng, "A Privacy-Preserving Federated Learning for Multiparty Data Sharing in Social IoTs," *IEEE Trans Netw Sci Eng*, vol. 8, no. 3, pp. 2706–2718, 2021, doi: 10.1109/TNSE.2021.3074185.

[17] Z. Ma, L. Wang, and W. Zhao, "Blockchain-Driven Trusted Data Sharing With Privacy Protection in IoT Sensor Network," *IEEE Sens J*, vol. 21, no. 22, pp. 25472–25479, 2021, doi: 10.1109/JSEN.2020.3046752.

[18] J. Chang, J. Ni, J. Xiao, X. Dai, and H. Jin, "SynergyChain: A Multichain-Based Data-Sharing Framework With Hierarchical Access Control," *IEEE Internet Things J*, vol. 9, no. 16, pp. 14767–14778, 2022, doi: 10.1109/JIOT.2021.3061687.

[19] H. Xie, J. Zheng, T. He, S. Wei, and C. Hu, "TEBDS: A Trusted Execution Environment-and-Blockchain-supported IoT data sharing system," *Future Generation Computer Systems*, vol. 140, pp. 321–330, 2023, doi: https://doi.org/10.1016/j.future.2022.10.016.

[20] T. Wang, Q. Wu, J. Chen, F. Chen, D. Xie, and H. Shen, "Health data security sharing method based on hybrid blockchain," *Future Generation Computer Systems*, vol. 153, pp. 251–261, 2024, doi: https://doi.org/10.1016/j.future.2023.11.032.

[21] G. Xiaofeng, L. Ying, and S. X. Sun, "Federated Content Rights Management for Research and Academic Publications Using the Handle System," *D-Lib Magazine*, vol. 16, no. 11/12, Nov. 2010, doi: 10.1045/november2010-guo.

[22] T. Weigel, S. Kindermann, and M. Lautenschlager, "Actionable Persistent Identifier Collections," *Data Sci J*, vol. 12, pp. 191–206, Jan. 2014, doi: 10.2481/dsj.12-058.

[23] M. J. Harvey, N. J. Mason, and H. S. Rzepa, "Digital Data Repositories in Chemistry and Their Integration with Journals and Electronic Notebooks," *J Chem Inf Model*, vol. 54, no. 10, pp. 2627–2635, Oct. 2014, doi: 10.1021/ci500302p.

[24] R. Quick, L. Lannom, M. Krenz, and Y. Luo, "E-RPID PEARC 2019: The Digital Object Architecture and Enhanced Robust Persistent Identification of Data," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, in PEARC '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–4. doi: 10.1145/3332186.3333255.

[25] L. Lannom, D. Koureas, and A. R. Hardisty, "FAIR Data and Services in Biodiversity Science and Geoscience," *Data Intell*, vol. 2, no. 1–2, pp. 122–130, Jan. 2020, doi: 10.1162/dint_a_00034.

[26] C. Sharp, "Overview of the digital object architecture (DOA)," 2016.

[27] S. Sun, L. Larry, and B. Brian, "RFC3650: Handle system overview," 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3650

[28] Li Xiao, Wang Feng, and Wei Mao, "Solving DNS Security Issues Using Handle Protocol Theory," *Application Research of Computers*, vol. 23, no. 12, pp. 104–107, 2006.

[29] M. A. Hisseine, D. Chen, X. Yang, and X. Wang, "Using Handle system to provide persistent identifiers for diploma," in *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 2022, pp. 68–73. doi: 10.1109/ICCECE54139.2022.9712843.

[30] U. Schwardmann, "Digital Objects – FAIR Digital Objects: Which Services Are Required?," *Data Sci J*, 2020, doi: 10.5334/dsj-2020-015.

[31] G. Xiaofeng and X. S. Sam, "The Development and Application of Handle System," *Digital library forum*, vol. 8, pp. 18–24, 2013.

[32] A. Hardisty, W. Addink, F. Glöckler, A. Güntsch, S. Islam, and C. Weiland, "A choice of persistent identifier schemes for the Distributed System of Scientific Collections (DiSSCo)," *Res Ideas Outcomes*, vol. 7, p. e67379, 2021, doi: 10.3897/rio.7.e67379.

BIOGRAPHIES OF AUTHORS

Mahamat Ali Hisseine received Bachelor's and master's degrees in computer science at the University of Science and Technology Beijing in 2012 and 2017 respectively. He is currently pursuing a Ph.D. degree in computer science at Tongji University. His research area includes Blockchain, Advanced Cryptographic Technologies in Blockchain, Smart Contracts Applications, data consistency, transparency and privacy in blockchain. Handle systems, Persistent Identifiers, and Industrial Internet of Things identifiers. He can be contacted at email: mahamat@tongji.edu.cn.

Prof. Deji Chen is a senior member, IEEE received a Ph.D. in computer science from the University of Texas, Austin, TX, USA, in 1999. He is presently a Professor at Wuxi University, in China. He has been involved in process automation for more than two decades. He took part in creating OPC and WirelessHART international standards, coauthored WirelessHART: Real-Time Mesh Network for Industrial Automation, and translated ISA bestselling books on process automation into Chinese. He is elevated to a distinguished grade of ISA fellowship. His research interests include Industrial Internet and 5G. He can be contacted at email: dejichen@tongji.edu.cn.

Yang Xiao received the Ph.D. degree in pattern recognition and intelligent systems from the Tongji University, Shanghai, China, in 2004. She conducted a Postdoctoral Research with the National University of Singapore, Singapore. She is an Associate Professor with the Department of Computer Science, School of Electronic and Information Engineering, Tongji University, since 2011. Her research interests include pattern recognition artificial intelligence information processing research and computer teaching. She can be contacted at email: xiaoyang@tongji.edu.cn.