Received: 12 July 2025 | Revised: 10 September 2025 | Accepted: 7 October 2025 | Published online: 16 December 2025

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2026, Vol. 5(2) 304–320
DOI: 10.47852/bonviewJCCE52026785

BON VIEW PUBLISHING

Sharded Fog Architecture for Scalable CP-ABE with Minimized Policy Trees in Blockchain Systems

Leon Wirz[1] and Pattarasinee Bhattarakosol[1,*]

[1] *Department of Mathematics and Computer Science, Chulalongkorn University, Thailand*

Abstract: The protection of Personally Identifiable Information (PII) remains a significant challenge, particularly within financial transactions. Current systems often fail to strike an optimal balance between security, performance, fine-grained access control, and distributed processing. A common dilemma is that some solutions prioritize security at the cost of efficiency, while others struggle with the complexity of managing detailed access controls in decentralized environments. Blockchain technology is widely recognized for its auditability, tamper resistance, and decentralization; however, the limitations of this method are computational efficiency of cryptographic operations and latency. Thus, to solve such limitations, this paper proposes an innovative approach that combines Ciphertext-Policy Attribute-Based Encryption (CP-ABE) with a lightweight fog layer. This hybrid design offloads resource-intensive cryptographic operations to the fog layer, effectively reducing the complexity of the policy tree while also protecting against potential side-channel attacks through a multi-layer encryption strategy. The proposed method not only strengthens data confidentiality but also ensures that the impact on transaction performance is minimal. This approach has been proven to be more secure, efficient, and a significantly distributed solution for managing sensitive information. Additionally, the research validates the effectiveness of this design through practical implementation and experimental comparisons with state-of-the-art systems, conducted across various settings. These experiments highlight the advantages of the proposed approach in terms of both security and operational efficiency regarding decryption and encryption, as experimental results show that this paper achieves near-constant encryption and decryption times at the Data Owner and Data User sides, reducing latency by over 80% compared to existing schemes.

Keywords: blockchain, fog computing, Attribute-Based Encryption, access control, IoT security, edge computing, lightweight cryptography

1. Introduction

Blockchain has emerged as a foundational technology for building decentralized and tamper-resistant systems [1], especially in industries that require trustless, transparent transactions. Its adoption is driven by the need to eliminate single points of failure and reduce reliance on centralized control [1, 2]. Despite its strengths, blockchain alone cannot guarantee full transactional confidentiality, particularly in environments dealing with sensitive data like financial records. While blockchain protects data integrity, it remains vulnerable to internal threats and node compromise, which can expose confidential information [1, 2]. This is critical when individual data points, though small, aggregate into large-scale, sensitive datasets. Protecting such data requires not only privacy but also fine-grained access control, ensuring that each participant accesses only what they are authorized to see.

Many existing privacy preserving techniques [3–6] focus on protecting content but neglect the need for selective access control. In financial systems, where multiple parties interact with data, enforcing such fine-grained policies is essential. However, implementing them efficiently on blockchain is challenging due to resource constraints and the high volume of transactions.

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [7] is one promising approach to enforce access control based on user attributes. Nevertheless, it suffers from high computational complexity, especially when policy trees grow deeper, making it unsuitable for real-time or large-scale applications. Some solutions offload computation [5] but often rely on unrealistic assumptions such as secure internal channels. Moreover, CP-ABE and traditional encryption schemes expose metadata and structural information in ciphertexts, which can be exploited through side-channel attacks [8]. This vulnerability becomes particularly dangerous in blockchain systems where transaction-level analysis is feasible. To address this, some works have explored outsourcing heavy computations [9–11], while others extend CP-ABE with revocation or multi-authority management [12–14]. Further studies integrate fog and blockchain to enhance privacy or distributed storage [15–17], and some incorporate intrusion detection and secure aggregation [18–20].

Parallel research emphasizes lightweight cryptography and system-level improvements. For example, privacy-preserving aggregation and blockchain-based access control schemes aim to reduce device-side costs [21–23], while hierarchical and dynamic models extend fine-grained control for IoT and smart grids [24–26]. At the infrastructure level, fog computing surveys highlight its potential for secure, distributed computation [27], while consortium blockchain models improve authorization and redactibility [28]. Recent studies further explore trust management and secure IoT data sharing in fog-assisted systems [29, 30]. Together, these works advance efficiency and decentralization, yet challenges remain in balancing security, scalability, and deployment feasibility.

Despite this progress, no existing solution comprehensively addresses all the requirements: confidentiality, access control,

***Corresponding author:** Pattarasinee Bhattarakosol, Department of Mathematics and Computer Science, Chulalongkorn University, Thailand. Email: Pattarasinee.B@chula.ac.th

scalability, and efficiency, in the context of financial transactions. This paper introduces a novel access control scheme that:

1) Integrate CP-ABE with a lightweight transaction encryption model specifically designed for financial systems, enabling secure, attribute-based access control while maintaining efficiency under high transaction volumes.
2) Offload computationally intensive cryptographic operations to a sharded fog-based architecture, which distributes the workload across decentralized nodes to significantly reduce processing burdens on client devices.
3) Optimize the CP-ABE decryption process by minimizing the policy tree using Boolean equation reduction techniques, thereby lowering both computational complexity and resource consumption.
4) Introduce a dual-layer encryption strategy that secures data at both the transaction and block levels using an aggregated CP-ABE master key, enhancing protection against metadata leakage and side-channel attacks within blockchain environments.

The rest of this paper is organized as follows: Section 2 reviews related work, Section 3 introduces the problem domain, Section 4 details the proposed scheme, Section 5 presents the security analysis, Section 6 reports experimental evaluation, Section 7 is a discussion on the results, and Section 8 concludes with future directions.

2. Literature Review

To safeguard sensitive data in decentralized systems such as fog computing and blockchain, extensive research has been conducted across multiple domains, including access control, data aggregation, encryption outsourcing, and privacy preservation. This literature review categorizes key contributions by the specific challenges they address, evaluating their approaches, limitations, and evolution over time.

Several studies have focused on fine-grained access control mechanisms tailored for fog-assisted and blockchain-integrated systems. Sarma et al. [9] proposed MACFI, a multi-authority CP-ABE scheme designed specifically for constrained IoT environments. Unlike traditional models where ciphertext and secret key sizes increase proportionally with the number of attributes, MACFI maintains a constant-size secret key and ensures that ciphertext growth depends only on the number of authorities. This greatly reduces the storage and bandwidth overhead for IoT devices, which is a critical requirement for resource-constrained applications. The scheme also offloads decryption to fog nodes, which helps alleviate computational burdens on end-user devices. However, although MACFI effectively addresses scalability and efficiency, it does not completely eliminate the linear growth of authority overhead. The need to manage and coordinate multiple authorities remains a challenge in large-scale IoT deployments, potentially leading to administrative complexity and performance degradation.

Building on CP-ABE integration with blockchain, Hou et al. [23] presented an attribute-based access control model that incorporates chameleon hashing and resistant-regeneration keys to enhance collusion resistance and tamper-proofing. By combining CP-ABE with blockchain immutability, the approach aims to strengthen both confidentiality and integrity of data sharing. The use of fog nodes to offload decryption helps reduce the computational load on end users, which is particularly beneficial for mobile or low-power devices. Nonetheless, the scheme only hides attribute values, while attribute names remain exposed, which can still leak contextual information and compromise user privacy. This partial privacy undermines the protection of sensitive metadata and may present a vulnerability in scenarios involving surveillance or data profiling.

The LIGHTMED framework, introduced in References [10, 11], similarly combines blockchain and fog computing for medical data access. It delivers auditable outsourced encryption and decryption for medical records, thereby improving efficiency on the user's side. This is especially crucial in healthcare settings where responsiveness and traceability are essential. However, it suffers from limitations such as the use of plaintext index storage on the blockchain, which exposes metadata to potential attackers. Additionally, the system does not adequately support superficial or temporary users who lack secure registration channels, limiting its usability in emergency scenarios or for secondary stakeholders, due to the lack of key revocation.

In the vehicular domain, Yang et al. [31] proposed a CP-ABE model for Internet of Vehicles (IoV) that incorporates reverse firewalls and multi-authority management to prevent data leakage and collusion. The model supports offline encryption and outsourced decryption, significantly reducing the computational load on vehicular terminals that often have limited processing capabilities. Reverse firewalls add an additional layer of protection by re-randomizing ciphertext during transmission, ensuring resilience against interception or tampering. However, the model introduces significant deployment complexity, particularly in coordinating multiple attribute authorities and equipping roadside units (RSUs) with reverse firewall functionality. These requirements could become logistical bottlenecks in large-scale or budget-constrained implementations.

A major line of research addresses the outsourcing of computationally expensive cryptographic operations to fog nodes to relieve the burden on end-user devices. Mu et al. [24] developed a lightweight data access control scheme for smart grids that offloads decryption to fog servers. This reduces the load on smart meters and enhances overall decryption performance. The model is particularly effective in environments where frequent data access is necessary, such as energy consumption monitoring. Nevertheless, while decryption is optimized, encryption costs remain moderately high, and the scheme lacks efficient attribute revocation, both of which could hinder long-term scalability and adaptability.

Shruti et al. [12] also targeted smart grid environments with a fog-assisted encryption model that balances efficiency, privacy, and resource constraints. By securely aggregating data from smart meters and forwarding it to the cloud, the model reduces transmission and storage overhead. However, reliance on a single pre-shared key introduces a single point of failure; if compromised, the entire system's security could be breached. Furthermore, the lack of fine-grained access control limits the ability to differentiate user permissions, which is essential in multi-stakeholder environments.

Peñuelas-Angulo et al. [13] designed a CP-ABE-based framework with broadcast encryption for Industrial IoT applications. The architecture incorporates multiple attribute authorities and leverages fog nodes to handle outsourced encryption and decryption, thereby optimizing bandwidth and computational usage. The scheme includes a revocation mechanism based on broadcast encryption, which enhances security by enabling dynamic user management. Nonetheless, the placement of CP-ABE operations on client devices poses performance risks for resource-constrained nodes, especially in dense industrial deployments with lightweight client resources.

Tao et al. [20] tackled decryption performance by introducing ORR-CP-ABE, which allows users with similar privileges to reuse decryption results, significantly reducing redundancy. The scheme utilizes a translation key to support compatibility across users and integrates Named Data Networking (NDN) to improve content retrieval efficiency. While this model demonstrates strong performance benefits in edge computing environments, it depends heavily on the willingness of users to contribute translation keys. In the absence of

proper incentives, this could degrade performance. Furthermore, the scheme still faces challenges related to complex policy tree evaluation, particularly on lower-performance devices.

Another key theme in the literature is secure storage and data integrity across fog and cloud environments. Ojha et al. [14] introduced a three-tier architecture using Hash-Solomon coding and intelligent data distribution across local, fog, and cloud layers. This model enhances fault tolerance and ensures that no single layer has access to all data segments. However, the communication channel between fog and local storage layers lacks explicit access control mechanisms, making it a potential attack surface for internal or lateral threats. The absence of security guarantees in this communication layer undermines the integrity of the architecture.

Dong et al. [28] contributed a Redactable Consortium Blockchain that offloads encryption and decryption via an external access control module. This design reduces the processing load on data owners and users, thereby improving usability. Yet, this approach introduces increased network traffic and cryptographic operation overhead, which may limit real-time performance and scalability. Additionally, managing the external module requires trust and availability assumptions that are not always guaranteed.

A similar balance between performance and security is addressed by dos Santos et al. [16], who developed IoTSafe, a fog-based platform integrating TLS, MQTT, and deep learning-based intrusion detection. The system is capable of secure data transmission and anomaly detection in real time, proving effective across multiple IoT environments. However, TLS imposes considerable cryptographic overhead during communication, especially in latency-sensitive applications, limiting its practical deployment in constrained networks.

Apat and Sahoo [17] advanced a fog-blockchain model using elliptic curve-based key exchange and Merkle trees for data verification. This model ensures secure communication and tamper-proof storage for biometric and medical data. While performance is favorable, the study primarily benchmarks against outdated cryptographic schemes such as RSA, limiting the relevance of its comparisons. Additionally, the model's scalability remains uncertain when dealing with a massive influx of IoT devices, which is a typical scenario in real-world deployments.

Zhang et al. [18] proposed EPri-MDAS, a homomorphic encryption-based aggregation scheme for smart grids that supports multidimensional data analysis and privacy preservation without relying on trusted authority. While the model improves data utility and fault tolerance, it does not adequately support lightweight external devices, which may restrict its implementation in household-level smart grid deployments.

In terms of system-level resilience, Feng et al. [15] introduced a reputation-based consensus mechanism tailored to fog computing. The approach uses differentiated metrics to assess physical devices and fog nodes, with periodic resets to prevent the buildup of falsely high reputations. This ensures fair consensus formation and accurate device trust evaluation. Still, the periodic reset mechanism may delay the identification of persistent attackers who operate just below detection thresholds.

Routray et al. [29] built a pairing-free CP-ABE scheme using elliptic curves and fog offloading to reduce encryption time in cyber-physical systems. The design supports decentralized key management and includes user and attribute revocation capabilities. However, despite the performance benefits, elliptic curve scalar multiplication remains a costly operation for devices with limited computational capacity, potentially offsetting the benefits of offloading.

Several works have explored the evolution of ABE-based schemes to include revocation, convergence, and dynamic policy handling. Sarma and Moulik [19] presented e-SAFE, a comprehensive CP-ABE framework optimized for medical IoT. It integrates attribute revocation, convergence, privileged access, and outsourcing while maintaining constant-size decryption keys. This combination makes the framework highly adaptive, but the practical feasibility of deploying such a complex feature set in real-world hospital environments still requires further validation.

Deng et al. [25] improved CP-ABE for hierarchical data by optimizing policy trees and grouping related policies, which reduces decryption overhead. This hierarchical structure is suitable for big data applications involving nested roles and permissions. However, the lack of outsourced decryption means that end-users are still responsible for heavy cryptographic operations, making it less viable for lightweight devices.

Further innovation is seen in work by Hundera et al. [26] who introduced HOOCLS-PRE, a certificateless signcryption system with proxy re-encryption for IoV environments. By dividing encryption into offline and online phases, the scheme ensures that most heavy operations occur in advance, thereby improving online efficiency. It enhances privacy by preventing unauthorized identity and location linking. Despite these advantages, the system's complexity, particularly in coordinating the dual cryptographic environments (CLC and IBC), may hinder its widespread adoption.

Finally, Narla et al. [30] developed a secure transmission framework for IoT data using clustering algorithms, Gauss Montgomery Curve Cryptography, and Merkle tree verification. This system leverages multiple layers of optimization for data encryption, query handling, and integrity checks. However, the cryptographic methods used are resource-intensive and may not be scaled effectively in large, real-time IoT deployments. Moreover, the scheme lacks a detailed analysis of how it manages dynamic device participation or query prioritization under load.

Together, these studies offer an evolving landscape of techniques aimed at improving the security, efficiency, and scalability of fog and blockchain-integrated data systems. While progress has been made, ongoing work is still required to address unresolved issues such as fine-grained privacy, lightweight support, and scalable access control. Table 1 presents a comparison of the objectives, methodologies, and application domains of key publications:

In summary, existing schemes fall short in addressing several critical aspects, including fine-grained access control granularity, full outsourcing of heavy cryptographic operations, multi-layer encryption for mitigating side-channel attacks, high availability, and policy tree optimization. These limitations highlight the need for a comprehensive approach that integrates these features to enhance both security and efficiency.

3. Problem Domain

In the realm of secure data management, particularly in high-stakes environments such as financial systems, ensuring fine-grained and efficient access control has become a pressing need. Financial transactions are characterized by their high throughput and sensitive nature, requiring robust protection mechanisms that do not compromise system performance. This dual requirement, strong security and low computational overhead, has prompted a surge in research and development efforts aimed at creating optimized access control architectures. However, despite significant progress, existing methodologies still present notable limitations that hinder their applicability in practical, high-performance environments.

Firstly, many of the current models have been developed with general-purpose use cases in mind and are not explicitly tailored to address the specific constraints of financial transaction systems. These systems demand not only high security and reliability but also real-time processing capabilities and scalability to handle massive

Table 1
Literature review comparison

Reference	Year	Author	Objectives	Approach	Application domain	Shortcomings
[21]	2020	Shen et al.	Confidentiality and efficiency	Fog computing	IoT	- Supports only single receiver. - Lack of fine-grained access control.
[22]	2021	Qin et al.	Confidentiality and efficiency	Blockchain	IoT	- Index storage as plaintext.
[9]	2022	Sarma et al.	Efficiency and storage size	Fog computing	IoT	- Cryptographic operations scale with Aas.
[10, 11]	2023	Fugkeaw et al.	Confidentiality and efficiency	Blockchain & fog computing	Healthcare IoT	- Index storage as plaintext. - Secure channel assumption.
[23]	2023	Hou et al.	Confidentiality and collusion resistance	Blockchain & fog computing	Healthcare	- Limited Blockchain usage. - Attribute hiding still conveys attribute names.
[24]	2023	Mu et al.	Confidentiality and efficiency	Server outsourcing	Smart Grid	- High encryption cost.
[12]	2024	Rani et al.	Confidentiality and efficiency	Fog computing	IoT	- Lack of fine-grained access control. - Single preshared key usage.
[25]	2024	Deng et al.	Confidentiality and efficiency	Policy tree optimization	Healthcare	- Lack of outsourcing - High encryption & decryption cost.
[13]	2024	Peñuelas-Angul et al.	Confidentiality and efficiency	Fog computing	IoT	- High cryptographic operation cost.
[14]	2024	Ojha et al.	Confidentiality and efficiency	Fog computing	IoT	- Unsecure channel between fog and local storage.
[15]	2024	Feng et al.	Confidentiality, authentication, and efficiency	Blockchain & fog computing	Smart City IoT	- Cyclical updates of reputation vulnerability.
[16]	2024	dos Santos et al.	Confidentiality and efficiency	MQTT & fog computing	IoT	- High cryptographic operation cost.
[26]	2024	Hundera et al.	Confidentiality, integrity, and efficiency	Proxy re-encryption	IoV	- High cryptographic operation cost.
[17]	2024	Apat and Sahoo	Confidentiality, integrity, and efficiency	Fog computing	Healthcare IoT	- Cryptographic operations scale with IoT devices.
[18]	2024	Zhang et al.	Confidentiality, availability, and efficiency	Fog computing	Smart Grid	- High decryption cost. - Data user is a centralized provider.
[19]	2024	Sarma and Moulik	Confidentiality, user revocation, and efficiency	Fog computing	Healthcare IoT	- High cryptographic operation cost
[20]	2024	Tao et al.	Confidentiality and efficiency	Server outsourcing, caching, & user grouping	IoT	- High cryptographic operation cost.
[28]	2024	Dong et al.	Confidentiality, redaction, and efficiency	Outsourcing	Organizations	- Increased network traffic overhead.
[29]	2025	Routray et al.	Confidentiality and efficiency	Fog computing	IoT	- High cryptographic operation cost.
[30]	2025	Narla et al.	Confidentiality, authentication, and efficiency	Fog computing	IoT	- Missing scalability indication. - High cryptographic operation cost.
[31]	2025	Yang et al.	Confidentiality, collusion resistance, and efficiency	Fog computing	IoV	- Practical implication due to deployment overhead of CRF-equipped RSUs.

volumes of transactions. The failure to align encryption strategies with such demanding operational conditions often results in bottlenecks and system slowdowns, undermining their effectiveness in real-world deployments.

Secondly, most solutions rely on computationally intensive encryption and decryption schemes such as attribute-based encryption (ABE), which, while powerful in terms of fine-grained control, tend to scale poorly with data size and attribute complexity. This poses a major challenge when deployed on end-user devices or edge nodes, which typically have limited processing power and memory. In financial applications, where latency and responsiveness are critical, such computational burdens can significantly degrade user experience and system responsiveness.

Lastly, although the foundational cryptographic algorithms used in many existing schemes are secure under standard attack models, they often fall short in addressing more sophisticated threats such as man-in-the-middle attacks, replay attacks, and side-channel attacks. These threats are particularly concerning in financial environments, where adversaries are often well-resourced and motivated. Without incorporating additional layers of defense, such as dual-layer encryption, randomized sharding, or secure multi-party computation. These systems remain vulnerable to breaches that can result in severe financial and reputational damage.

Given these limitations, there remains a clear need for a security framework that not only meets the rigorous access control demands of financial transactions but also minimizes computational cost and fortifies the system against advanced attack vectors. This gap serves as the foundation for further exploration, as detailed in the subsequent Literature Review section, where existing solutions and their shortcomings are systematically examined to inform the development of a more suitable approach.

4. Proposed Scheme

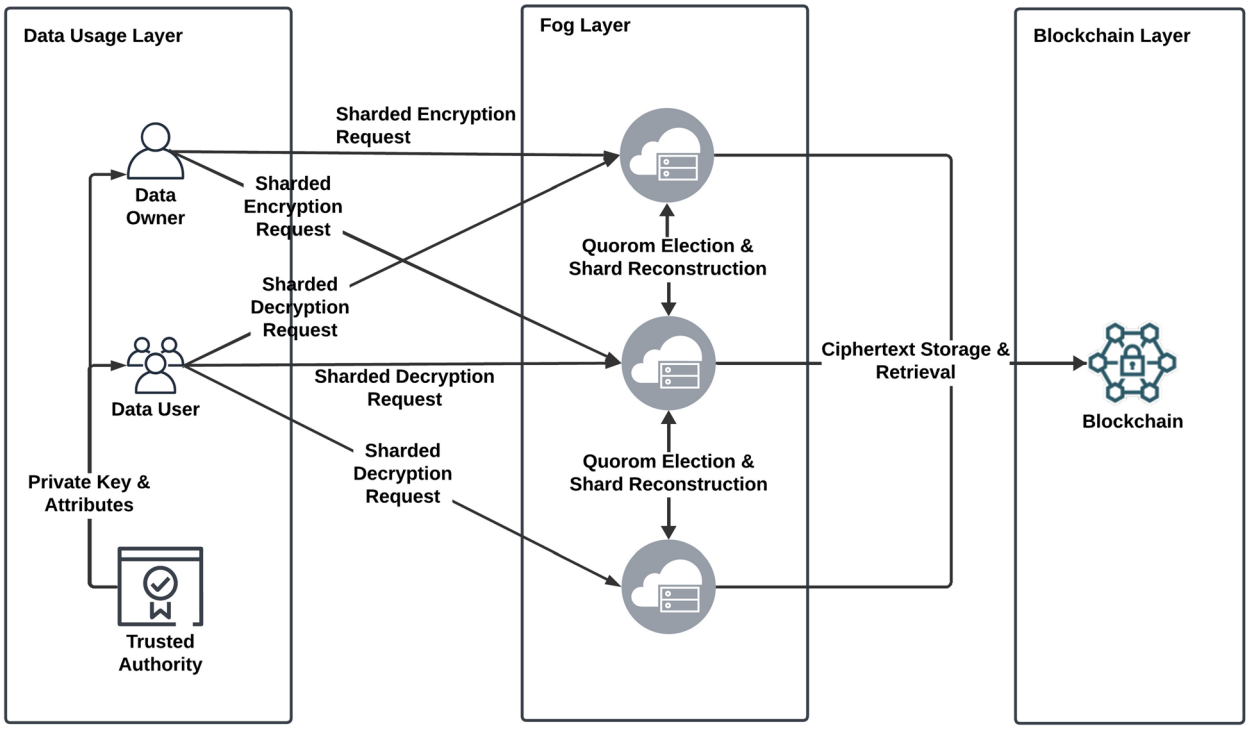
This section will cover the proposed scheme in detail, which includes both a thorough walkthrough of the architecture and cryptographic construct.

4.1. System architecture

The system architecture comprises five different types of entities: Blockchain, Fog Nodes, Data Owners, Data Users, and Trusted Authority, as shown in Figure 1. Details of each component are described below.

1) Blockchain: The Blockchain in this scheme serves as a decentralized ledger for storing encrypted transaction data and as an execution layer for smart contracts that manage access control and audit functions. A Proof of Authority (PoA) consensus mechanism is used to ensure fast finality and low overhead, which is well-suited for permissioned, high-throughput environments like financial systems. Compared to alternatives like PBFT or PoS, PoA offers better scalability with reduced complexity, though it relies on a fixed set of trusted validators, making it ideal for controlled, consortium-based deployments.
2) Fog nodes: Fog nodes are responsible for encrypting incoming plaintext data and reconstructing the original format from sharded decryption requests. The system assumes a distributed fog infrastructure composed of a larger number of lightweight nodes, promoting scalability and efficiency at the network edge.
3) Data owners: Data owners are the primary sources of information and initiators of encryption requests. They are tasked with defining the attribute sets associated with each request and distributing the data across the system in a sharded and encrypted manner.

Figure 1
System architecture



4) Data users: Data users are the recipients of information, who issue sharded decryption requests to the fog layer in order to retrieve and reconstruct the intended content, subject to access policy verification.
5) Trusted Authority (TA): The Trusted Authority acts as the central entity for system initialization, responsible for generating global parameters, cryptographic keys, and attribute sets. It is assumed to operate in a secure and incorruptible environment.

4.2. Cryptographic construct

Our cryptographic construct contains four processes: System Initialization, Key Generation and Distribution, Data Encryption, and Data Decryption. Table 2 explains the notations used for all of our processes.

4.2.1. Phase 1: system initialization

The initial phase of the proposed scheme comprises two distinct functions: *initSystem*, invoked globally by the Trusted Authority to initialize system-wide parameters, and *genSymKeyFunc*, executed independently by each fog node and the Blockchain to generate local symmetric key functions.

Table 2
Notation description

Symbol	Description
TA	Trusted Authority
κ	Secure parameter used for Trusted Authority Initialization
DO_E	Data Owner identified by E
DU_E	Data User identified by E
PK	Public key generated by the TA
MK	Master key generated by the TA
symKeyFunc	Function to generate a Symmetric key
symKeySeed	Seed value to generate a Symmetric key
symKey	Symmetric Key
SAE	Attribute Collection assigned to entity E
SKE	CP-ABE Secret Key of Entity E
PT	Plaintext of Data
CTE	Ciphertext of entity E
T	Policy Tree used in CP-ABE
NFG	Number of Fog Nodes sharded requests must be sent to.
cT	Sharded Ciphertext
t	Timestamp, with the following format: yyyy-MM-dd HH:mm:ss. SSSSSS
IDE	Identifier of Entity E
iCT	Intermediate Ciphertext, typically not secure and easy to break
φ	Chunk size of splitting up iCT to Fog Nodes
tmpK	Temporary key generated at fog layer
aggiCT	Aggregated intermediate ciphertext, by collecting each Sharded Ciphertext from each fog Node
aggtmpK	Aggregated Temporary key, by collecting each Temporary key from each fog node
BE	Blockchain Block E
nC	Hashmap containing fog node to temK mappings

1) initSystem(κ): (PK, MK)

The first method is initiated by providing a single secure parameter κ as input and returns two outputs: the public key (*PK*) and the master key (*MK*). This process leverages bilinear map operations [7] for secure parameter generation. Specifically, the method selects a bilinear group G_0 of prime order p, and randomly samples two values $\alpha, \beta \in Z_p$. The public key (*PK*) is then derived using these values as follows:

$$PK = \left\{ G_0, g, h = g^{\beta}, f = g^{\frac{1}{\beta}}, e(g,g)^{\alpha} \right\} \tag{1}$$

While the *MK* is being proven as:

$$MK = (\beta, g^{\alpha}) \tag{2}$$

2) genSymKeyFunc(): symKeyFunc

To enhance security, the symmetric key is not stored directly; instead, it is dynamically generated through a dedicated function. This function utilizes a cryptographically secure pseudo-random number generator (CSPRNG) to produce a random value, which is then XORed with a second independently generated random value, also derived via CSPRNG. Conceptually, this approach can be viewed as splitting the symmetric key into two securely generated components: *symKeyFunc* and *symKeySeed*. This can be seen in Algorithm 1.

Algorithm 1
Generate symmetric key generation function

Algorithm 1 genSymKeyFunc

```
genSymKeyFunc
    randValue ← CSPRNG.generateRandomValue()
    Fn(val) ← for i ← 0 to randValue.length()-1 do
        | val[i] ⊕ randValue[i];
    return Fn
```

4.2.2. Phase 2: key generation and distribution

At this stage of the scheme, two primary functions are executed: *genDataOwnerKey* and *genDataUserKey*, invoked by the *DO* and *DU*, respectively.

1) genDataOwnerKey(PK, MK, SA_{DO}): SK_{DO}

This function takes three input parameters: the public key (*PK*) and master key (*MK*), which are generated by the Trusted Authority during system initialization, and SA_{DO}, the attribute set assigned to the *DO*. The function outputs SK_{DO}, the secret key assigned to the Data Owner, which is subsequently used for decrypting ciphertext files. The generation of SK_{DO} involves selecting a random value $r \in Z_p$, and for each attribute j in SA_{DO}, a corresponding random value $r_j \in Z_p$ is independently sampled. This leads to the following mathematical formulation:

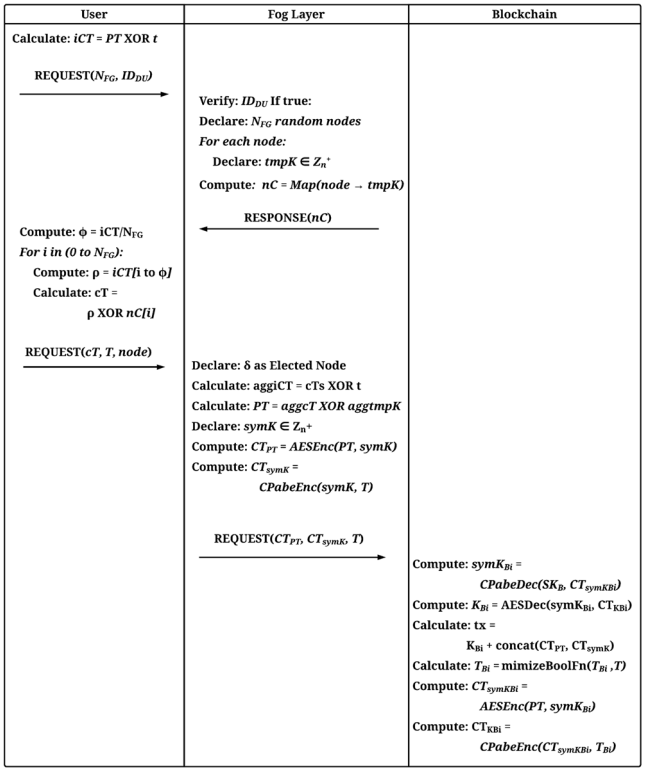
$$SK_{DO} = \left(D = g^{\frac{\alpha+r}{\beta}}, \forall j \in S : D_j = g^r \cdot H(j)^{rj}, D'_j = g^{rj} \right) \tag{3}$$

2) genDataUserKey(PK, MK, SA_{DU}): SK_{DU}

The same procedure as in the *genDataOwnerKey* method is applied here, with the distinction that SK_{DO} is replaced by SK_{DU}.

Subsequently, each generated key is securely stored on the respective devices of the *DO* and *DU*, with no reliance on any intermediate or external storages for either temporary or permanent key retention.

Figure 2
Encryption process diagram



4.2.3. Stage 3: data encryption

The principle flow of this stage can be seen in Figure 2, which shows how each entity interacts. At this stage of the scheme, four primary functions are executed: shardRequest is executed by the DO, electQuorum and encryptFogare executed by fog nodes, and encryptBlockchain is executed by the Blockchain.

1) shardRequest(PT, t, φ, N_{FG} , nC): Collection[cT]

The function, as can be seen in Algorithm 2, involves preparing to dispatch plaintext and attribute data to the nearest fog nodes. A timestamp t is incorporated to ensure that each shard request is unique, effectively mitigating the risk of replay attacks and preventing redundant transmission of identical data. Once this pairing is constructed, the *DO* proceeds to issue shard requests to the N_{FG} closest fog nodes. During this phase, the *DO* contacts each of the selected fog nodes with a unique request. The request carries all necessary elements to facilitate partial encryption at the fog layer, including the policy tree T, the number of fog nodes N_{FG}, an offset ϕ to introduce controlled randomness, and temporary node secrets nC for added obfuscation. The intermediate ciphertext iCT is computed by XORing the plaintext PT with t, providing a base layer of temporal randomness and uniqueness.

Algorithm 2
Initiate shard request

Algorithm 2 shardRequest

```
shardRequest (PT, t, ϕ, N_FG, nC)
    returnCollection ← newCollection();
    iCT ← PT ⊕ t;
    for i ← 0 to N_FG do
        temp ← iCT.get[i to i + ϕ];
        cT ← temp ⊕ nC[i];
        returnCollection.append(cT);
    return returnCollection
```

Ultimately, each fog node receives its tailored shard, along with the context necessary for the subsequent encryption phase.

2) electQuorum(ID_{Req}, CT, N_{FG}): node

Following the distribution of ciphertext shards to the fog nodes, a quorum must be established to facilitate the reconstruction of the original intermediate ciphertext. The *electQuorum* function is responsible for this selection process, aiming to designate a leader node among the set of fog nodes that participated in the previous sharding phase. At the core of this mechanism, each node initiates a listening thread through a dedicated socket, enabling communication within the local cluster. The process begins by generating a hash map to store shard-related data and identifying adjacent nodes that are part of the current context, typically determined using a request identifier ID_{Req}. Each node then enters a loop where it either continues broadcasting its state or listens for messages from neighboring nodes. Until the number of received shards equals the required threshold N_{FG}, each node actively disseminates its shard to others using a SHARD message. Once a node detects that the quorum size has been reached, it assumes the leadership role, broadcasts a COMPLETE message to its peers, and designates itself as the elected node. Throughout this process, all nodes monitor incoming messages. If a COMPLETE message is received from another node, that node is accepted as the elected leader. Otherwise, shard data from other nodes is recorded into the hash map for quorum validation. Algorithm 3 illustrates the pseudo code of this functionality.

Algorithm 3
Quorum election

Algorithm 3 electQuorum

```
electQuorum (ID_Req, CT, N_FG)
    cTHashMap ← newHashMap();
    electedNode ← null;
    nodes ← adjacentNodes.getInCurrentProcess(ID_Req);
    socket = quorumPort.listen();
    while electedNode == null do
        if cTHashMap.size() ≤ N_FG − 2 then
            foreach node ∈ nodes do
                sendFogMsg(CT, "SHARD", node);
        else
            foreach node ∈ nodes do
                sendFogMsg(null, "COMPLETE", node);
            electedNode = this();
            break
        payload = socket.read();
        if payload.message == "COMPLETE" then
            electedNode = payload.sourceNode
        else if payload.message == "SHARD" then
            cTHashMap[node] = payload.ctValue
        wait(100)                          ▷ Wait 100ms for retry
    return electedNode
```

3) encryptFog(CTs, t, MK, T, nC, symKeyFunc): (CT_{PT}, $CT_{symKeySeed}$)

Once the elected node has successfully collected all temporary keys and ciphertext fragments from the participating fog nodes, it begins the Data Encryption Stage. This phase is critical, as it reconstructs the original plaintext PT from the distributed ciphertext fragments *CTs* and the randomized secret key material. The first step involves deriving a shared symmetric encryption key. A CSPRNG is used to produce a random seed *randValue*. This seed serves as the basis for generating the actual symmetric key through *symKeyFunc*, ensuring that the key is tightly bound to the current encryption context. Afterwards, the original plaintext is reconstructed by XORing all received ciphertext fragments *CTs* with the corresponding temporary fog node secret values nC and t. This aggregated plaintext PT is then encrypted using the previously

derived symmetric key via a standard AES encryption scheme, resulting in the symmetric layer ciphertext CT_{PT}. To enable fine-grained access control, the random seed used to generate the symmetric key is itself encrypted using Ciphertext-Policy Attribute-Based Encryption (CP-ABE). This encryption takes as input the access policy T, the master key MK, and the random seed randValue, producing $CT_{symKeySeed}$. Finally, the function returns a tuple consisting of the encrypted plaintext CT_{PT} and the encrypted symmetric key seed $CT_{symKeySeed}$, completing the fog-level encryption process. The pseudocode for this process can be found in Algorithm 4.

Algorithm 4
Encryption on fog nodes

Algorithm 4 encryptFog

encryptFog *(CTs, t, MK, T, nC, symKeyFunc)*
- *randValue* ← CS PRNG.*generateRandomValue*()
- *PT* ← *CTs* ⊕ *nC.values* ⊕ *t*;
- *symKey* ← *symKeyFunc*(*randValue*);
- CT_{PT} ← *AES*.Encrypt(*PT*, *symKey*);
- $CT_{symKeySeed}$ ← *CPABE.Encrypt*(*randValue, MK, T*);
- **return** $(CT_{PT}, CT_{symKeySeed})$

4) encryptBlockchain(CT_{PT},$CT_{symKeySeed}$, T, MK)

In the final phase of the Data Encryption Stage, as can be seen in Algorithm 5, the encrypted output is securely stored on the blockchain to ensure immutability and traceability. This is facilitated by the *encryptBlockchain* function, which accepts the symmetric-layer ciphertext CT_{PT} and the CP-ABE-encrypted symmetric key seed $CT_{symKeySeed}$, along with the current access policy T and the blockchain master key MK. Upon invocation, the blockchain node first decrypts the latest block using its master key, a CP-ABE key instantiated with all valid attributes to inspect the current state of the blockchain. Following the decryption of the current block, the newly received ciphertexts are appended as a transaction. Each transaction contains both the encrypted data and its associated encrypted symKeySeed. To prevent potential conflicts where Data Users (*DUs*) may lack the necessary attributes to satisfy overly specific or incompatible policy trees, a policy optimization strategy is employed. All access structures T from transactions within a block are aggregated into a unified policy tree. This composite structure is then optimized using tree balancing and boolean expression minimization techniques as described in prior works [32, 33], to reduce redundancy and ensure minimal, yet expressive, policy definitions. This process enhances the efficiency of CP-ABE decryption during data access and ensures broader compatibility across authorized users. Once the ciphertexts are incorporated and the optimized access policy is attached, the block is finalized and committed to the consortium blockchain.

Algorithm 5
Encryption on blockchain

Algorithm 5 encryptBlockchain

encryptBlockchain $(CT_{PT}, CT_{symKeySeed}, T, MK)$
- *block* ← *this.getLatestBlock*()
- *CPABEKey* ← *this.getCPABEKey*()
- *symKeyFunc* ← *this.getSymKeyFunc*()
- $CT_{blockSymKeySeed}$ ← *this.getEncryptedSymKeySeed*()
- $CT_{blockData}$ ← *this.getEncryptedBlockData*()
- *originalT* ← *this.getCurrentPolicyTree*()
- *blockSymKeySeed* ← *CPABE.Decrypt*($CT_{blockSymKeySeed}$, *CPABEKey*);
- *symKey* ← *symKeyFunc*(*blockSymKeySeed*);
- *BlockData* ← *AES.Decrypt*($CT_{blockData}$, *symKey*);
- *newBlockData* ← *BlockData.append*((CT_{PT}, $CT_{symKeySeed}$))
- *newT* ← *originalT.merge*(*T*).*minimize*()
- $CT_{newBlockData}$ ← *AES.Encrypt*(*newBlockData, symKey*);
- $CT_{symKeySeed}$ ← *CPABE.Encrypt*(*symKeySeed, MK, newT*);
- *this.commitTransaction*($CT_{newBlockData}$, $CT_{symKeySeed}$)

4.2.4. Stage 4: data decryption

The last stage of the scheme includes six functions are executed: *shardRequest* and *decryptDataUser* are executed by the Data User (DU), *electQuorum* and *decryptFog* are executed by fog nodes, and *decryptBlockchain* is executed by the Blockchain. The interaction of each component can be seen in Figure 3, which gives a general overview of how each functionality is being executed.

1) shardRequest(SK_{DU}, t, φ, N_{FG} , nC): Collection[cT]

This follows the exact same logic as in *shardRequest* in the Data Encryption Stage, where the distinction lies withing the passed in *PT*, which is defined as the secret key of the *DU* SK_{DU} instead.

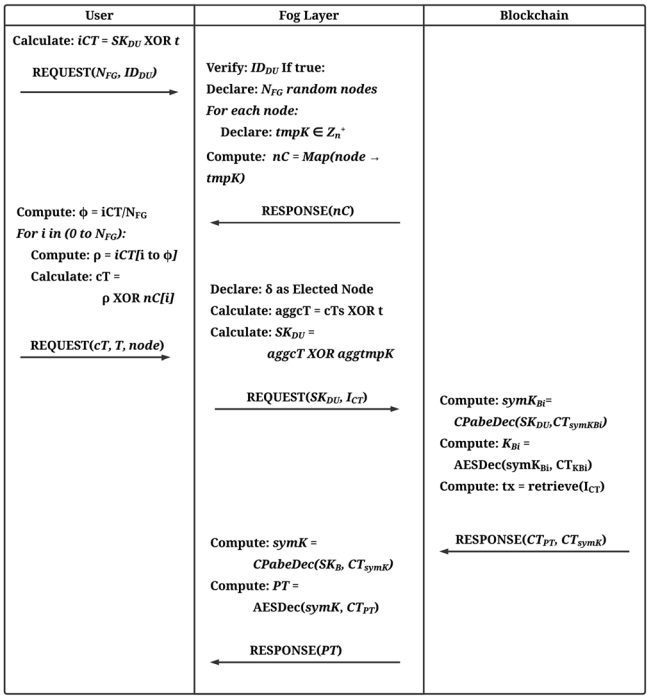
2) electQuorum(ID_{Req}, CT, N_{FG}): node

The *electQuorum* function is the same as in the Data Encryption Stage, where again the difference lies withing the passed in CT of the PT, which is defined as the CT of SK_{DU} instead. Afterwards, the fog node makes a request to the Blockhchain for the transaction retrieval.

3) decryptBlockchain(Req,SK_{DU}): (CT_{PT},$CT_{symKeySeed}$)

The *decryptBlockchain* function, as in Algorithm 6, initiates the Data Decryption Phase by enabling a DU to retrieve encrypted content from the blockchain. It accepts a request object Req, containing metadata such as the transaction ID, and the user's secret CP-ABE key SK_{DU}, which encapsulates the *DU's* access attributes. Upon invocation, the function first identifies the target transaction by extracting its ID from the request and fetching the corresponding block from the blockchain. To recover the block symmetric key used in the encryption process, the node retrieves the CP-ABE-encrypted block symmetric key seed from the block. This seed, encrypted under a policy defined

Figure 3
Decryption process diagram



during data encryption, is decrypted using the user's secret key SK_{DU}. This decryption process verifies whether the *DU* satisfies the access structure embedded in the ciphertext, only if the attributes in SK_{DU} fulfill the policy will the symmetric key seed be successfully recovered. The decrypted block seed is then passed through a predefined key derivation function *symKeyFunc*, generating the block symmetric key used to encrypt the original block. Using this symmetric key, the function proceeds to decrypt the ciphertext, producing the plaintext content of the block. Finally, *decryptBlockchain* returns both the symmetric-layer ciphertext CT_{PT} and the CP-ABE-layer $CT_{symKeySeed}$ to the elected fog node.

Algorithm 6
Decryption on blockchain

Algorithm 6 decryptBlockchain

decryptBlockchain *(Req, SK_{DU})*
- *transaction* ← *Req.getRequestedTransactionID*()
- *symKeyFunc* ← *this.getSymKeyFunc*()
- $CT_{blockData}$ ← *this.getBlockFromTransaction(transaction)*
- $CT_{blockSymKeySeed}$ ← *this.getEncryptedSymKeySeed*()
- *blockSymKeySeed* ← *CPABE.Decrypt*($CT_{blockSymKeySeed}$, SK_{DU});
- *symKey* ← *symKeyFunc*(*blockSymKeySeed*);
- *BlockData* ← *AES.Decrypt*($CT_{blockData}$, *symKey*);
- (CT_{PT}, $CT_{symKeySeed}$) = *BlockData.getTransaction*(*transaction*)
- **return** (CT_{PT}, $CT_{symKeySeed}$)

4) decryptFog($CT_{SK_{DU}}$, t, nC, CT_{PT}, $CT_{symKeySeed}$, symKeyFunc): PT

The *decryptFog* function performs the core data decryption process at the fog layer. This function is invoked with the CP-ABE-encrypted *DU* secret key $CT_{SK_{DU}}$, a time reference *t*, the temporary fod node secrets *nC*, the symmetric-layer ciphertext CT_{PT}, the CP-ABE-encrypted symmetric key seed $CT_{symKeySeed}$, and the symmetric key generation function *symKeyFunc*.

On the function call, the fog node first reconstitutes the time-bound decryption key SK_{DU} by XORing the encrypted key $CT_{SK_{DU}}$ with the values of *nC* and *t*. This ensures that the decryption key is valid only for a specific time window and resists replay or misuse outside its designated validity. Next, the fog node attempts to decrypt the CP-ABE-encrypted symmetric key seed $CT_{symKeySeed}$ using the derived SK_{DU}. This operation enforces fine-grained access control by verifying that the *DU's* attributes satisfy the policy under which the seed was encrypted. If the policy conditions are met, the original key seed is recovered and passed through the *symKeyFunc*, which derives the actual symmetric key used during encryption. Using this symmetric key, the fog node proceeds to decrypt the ciphertext CT_{PT} via symmetric decryption (e.g., AES), thereby recovering the original plaintext *PT*. Algorithm 7 explains this process via pseudocode.

Algorithm 7
Decryption on fog nodes

Algorithm 7 decryptFog

decryptFog *($CT_{SK_{DU}}$, t, nC, CT_{PT}, $CT_{symKeySeed}$, symKeyFunc)*
- SK_{DU} ← $CT_{SK_{DU}}$ ⊕ *nC.values* ⊕ *t*;
- *symKeySeed* ← *CPEABE.Decrypt*($CT_{symKeySeed}$)
- *symKey* ← *symKeyFunc*(*symKeySeed*);
- *PT* ← *AES.Decrypt*(CT_{PT}, *symKey*);
- **return** *PT*

5) respondShardRequest(ID_{Req}, t, nC, PT, N_{FG})

The *respondShardRequest* function, as can be seen in Algorithm 8, governs the decentralized response to data shard requests in a secure and distributed manner. When invoked, the function begins by identifying the node designated to lead the current response process through a leader election mechanism as elected from the electQuorum function. This leader is responsible for coordinating the sharding and propagation process. Using the request identifier ID_{Req}, the system retrieves all adjacent nodes participating in the current process instance using the context. If the elected node is the current node, the plaintext is first obfuscated using the temporary fog node secrets *nC* and *t* via XOR operation to produce a temporary cipher text. This intermediate is then divided into fragments using a threshold-based secret sharing mechanism by a parameter φ, which is derived from the size of N_{FG}.The resulting ciphertext fragments *cT* are indexed and distributed among the participating nodes.

Algorithm 8
Initiate shard response

Algorithm 8 *respondShardRequest*

respondShardRequest *(ID_{Req}, t, nC, PT, N_{FG})*
- *electedNode* ← *this.getElectedNode*();
- *nodes* ← *adjacentNodes.getInCurrentProcess*(ID_{Req});
- **if** *electedNode* == *this*() **then**
  - *temp* ← *PT* ⊕ *t* ⊕ *nC*;
  - *ϕ* ← *cT*/N_{FG}
  - **foreach** *index, node* ∈ *nodes* **do**
    - *cT* ← *temp.get*[*index to index* + *ϕ*];
    - **if** *index* == 0 **then**
      - *sendShardToDU(cT)*
    - **else**
      - *node.respondShardRequest*(ID_{Req}, *t, nC, cT,* N_{FG});
- **else**
  - *sendShardToDU(PT)*

6) decryptDataUser(CTs, t, nC): PT

Finally, in Algorithm 9, the user then calls the *decryptDataUser* function, which takes in all the sharded fragments *CTs*, *t*, and *nC*. When calling, the *CTs* are XORed with *t*, and *nC* yielding the requested Data PT.

Algorithm 9
Decryption at data user

Algorithm 9 decryptDataUser

decryptDataUser *(CTs, t, nC)*
- PT ← *CTs* ⊕ *t* ⊕ *nc*;
- **return** *PT*

5. Security Analysis

This section presents a formal security analysis of the proposed system, including the underlying security model and relevant security properties.

5.1. Security model

We adopt a game-based approach to prove the security of the proposed scheme. The central cryptographic primitive employed for secure data storage is Ciphertext-Policy Attribute-Based Encryption (CP-ABE), as originally introduced by Bethencourt et al. [7].

In the proposed system model, data is stored on a blockchain, while decryption keys are delegated to fog nodes. Data Owners (DOs) are treated as fully trusted, whereas Data Users (DUs) are only semi-

trusted. Furthermore, we assume that adversaries may make adaptive key extraction queries and that the Trusted Authority can be statically compromised. The security of CP-ABE is analyzed based on a standard interaction between an adversary (denoted A) and a challenger (denoted C).

Theorem 1

If no polynomial-time adversary exists that can break the security of CP-ABE with non-negligible advantage, then the proposed scheme is also secure against such adversaries under the same assumption.

1) Setup phase

A key pair is generated by the Trusted Authority (TA) and sent to the simulator, which also creates a separate auxiliary key pair. The challenger then executes *initSystem* for each trusted authority in the set *SA-S'A*, and provides the public key *PK* to adversary *A*. For each compromised authority in *S'A*, both the public and secret keys are disclosed to *A*.

Phase I

The adversary submits a set of attributes S to the challenger, which are associated with trusted authorities. In return, the challenger provides the corresponding secret keys for those attributes to adversary A.

2) Challenge phase

The adversary provides two messages m_0 and m_1 of equal length to the simulator. A random bit $\beta \in \{0,1\}$ is selected, and the message m_β is encrypted into a ciphertext using the CP-ABE scheme:

$$CT = \left(T, C'' = m_\beta \cdot Z, CT = h^S, \forall y \in: C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\right) \quad (4)$$

a. If $\mu = 0$, then $z = e(g,g)^{\alpha s}$, making the ciphertext a valid encryption of m_β.
b. If $\mu = 1$, then $z = e(g,g)^z$, resulting in $C'' = m_\beta \cdot e(g,g)^z$, which appears completely random to A and leaks no information about m_β.

Phase II

The same procedure from Phase I is repeated.

Finally, Adversary A makes a guess β' for the hidden bit β. The adversary's advantage is defined as:

$$Adv = \Pr[\beta = \beta'] - \tfrac{1}{2} \quad (5)$$

The scheme is considered secure if any probabilistic polynomial-time adversary can achieve only a negligible advantage.

Theorem 2

No adversary can recover the encrypted message using only the partial decryption obtained from a compromised fog node.

3) Decryption phase

If adversary A gains access to the partial ciphertext decrypted using CP-ABE, they still cannot retrieve the full plaintext. To recover the AES-encrypted data, A must obtain:

a. The key derivation function of the Data Owner (*symKeyFunc*)
b. The second partial key, which is securely stored on the blockchain and managed through smart contracts, is responsible for authentication and audit logging.

Theorem 3

If no adversary can distinguish between a partial value δ_p and its permutation δ'_p within polynomial time, then the derived value v_p is computationally indistinguishable from v'_p.

4) Shard encryption phase

Consider a scenario where adversary A intercepts ciphertexts transmitted by or to Fog nodes. Each node is assigned to a unique random value derived from a permuted list δ'_p, which is a permutation of the original list δ_p. In order to correctly identify the position of the actual data transmitted by a specific device, since the ciphertext includes both real and dummy data concatenated, the adversary must determine the exact mapping of the device's value in the permutation δ'_p. Thus, to extract the actual value v_p from among the set of permuted values v'_p, adversary A must first succeed in distinguishing the permutation δ'_p from the original δ_p, a task assumed to be computationally infeasible.

5.2. Adversary model and threat classification

We define a threat model under the Dolev-Yao assumption. An adversary A ∈ PPT (probabilistic polynomial time) has the ability to:

1) Intercept, forge, and replay messages.
2) Gain partial access to compromised fog nodes.
3) Statistically or adaptively corrupt Data Users (DUs) or fog nodes.
4) Execute chosen plaintext (CPA) or chosen ciphertext (CCA) attacks under certain bounds.

1) Sybil attacks

Let G be the set of fog nodes and A be a malicious adversary that generates $t > 1$ identities $\{id_1, \ldots, id_t\}$. Each registration $r_i = H(id_i \parallel nonce_i)$ must satisfy unique constraints validated by smart contracts. Assuming H is a random oracle, the probability that A can generate $r_i = r_j$ for $i \neq j$ is negligible in λ i.e.,

$$\Pr[\exists i \neq j; r_i = r_j] \leq \text{negl}(\lambda) \quad (6)$$

2) Double voting or double spending

Let tx_i and tx_j be two transactions from the same sender within Δt where $tx_{i.timestamp} = tx_{j.timestamp}$. The smart contract enforces that:

If $tx_{i.sender} = tx_{j.sender} \wedge tx_{i.timestamp} = tx_{j.timestamp}$, then tx_j is rejected.

Therefore,

$$\Pr[\text{A successfully submits } tx_j \text{ after } tx_i] \leq \text{negl}(\lambda) \quad (7)$$

3) Denial of service (DoS) attacks

The workload is distributed across fog shards using randomized assignment $\pi : D \to \{F_1, \ldots, F_k\}$, where D is the set of data packets and F_i are fog nodes. Assuming each fog node can process β packets/second, and A sends $\alpha > \text{kβ}\ \alpha > k_\beta$ packets to a subset $F' \subseteq F$, the system triggers rerouting after τ seconds to a new π'. Thus,

$$\Pr[\text{DoS leads to persistent outage beyond } \tau] \leq \varepsilon \quad (8)$$

for small ε.

4) Man-in-the-middle (MitM) and replay attacks

All messages are timestamped t and encrypted using ephemeral session key $k \in K_E$ generated via Diffie-Hellman exchange:

$$Enc_k(\text{m} \parallel \text{t}) = \text{c} \quad (9)$$

Replay succeeds only if:

$$|t_{now} - \text{t}| < \delta \wedge \text{c} \notin \text{Cache} \quad (10)$$

With secure nonce and freshness enforcement, this probability is:

$$\Pr[\text{Replay accepted}] \leq \text{negl}(\lambda) \tag{11}$$

5) Side-channel attacks

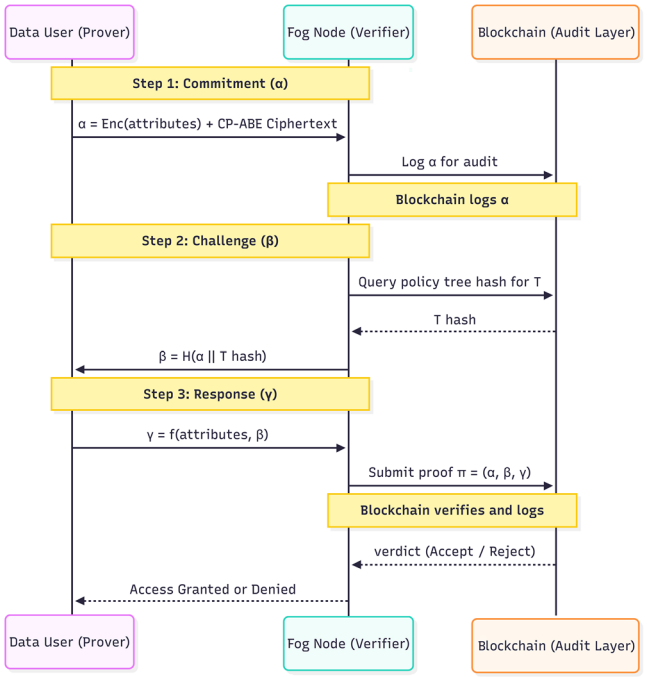
As explained in the Challenge Phase and Decryption Phase, CP-ABE prevents adversaries from distinguishing partial values from permutations within polynomial time and hence making the derivation of the final value indistinguishable. However, to improve upon this point this whole concept is also applied on a block-level. This results in no adversary can distinguish between a partial block B_p and its permutation B'_p within polynomial time, then the derived block Γ_p is computationally indistinguishable from Γ'_p. With this, even the underlaying partial value δ_p is hidden from any adversary and hence making it a multi-layer prevention scheme.

In practice, the adversary model extends naturally to resource-constrained IoT and edge environments, where devices are particularly vulnerable to compromise. The proposed architecture mitigates such risks by minimizing computational load at Data Owners and Data Users through lightweight XOR-based operations, while relegating intensive cryptographic tasks to fog nodes. Under the Dolev–Yao assumption, adversaries are prevented from forging or replaying valid transactions, while collusion resistance is ensured by CP-ABE's access structure enforcement. Even if a subset of fog nodes is compromised, adversaries cannot fully reconstruct plaintext data without satisfying access policies and obtaining blockchain-verified partial keys. This threat classification highlights that the scheme provides practical security for IoT deployment, while leaving denial-of-service resilience and physical device compromise as areas for future enhancement.

5.3. Zero-knowledge proof

To ensure privacy and correctness without revealing sensitive attribute values or decrypted results, a non-interactive zero-knowledge proof (NIZKP) is used during policy matching. The protocol operates as follows in Figure 4:

Figure 4
Zero-knowledge proof diagram



This protocol ensures that the prover has a valid set of attributes satisfying the access policy without revealing the actual attributes, and the verifier can validate the claim with zero knowledge of the data or attributes. It strengthens privacy while guaranteeing correctness of access control decisions.

5.4. Applicability in adversarial environments

While the proposed scheme is primarily designed for consortium-based and permissioned environments where fog nodes can be managed, monitored, and audited, it is important to consider how the architecture may hold up in less controlled or open settings. In such environments, adversaries may act beyond the semi-honest assumption, including malicious fog nodes that deviate from protocol execution, colluding Data Users, or large-scale Sybil attacks. Although the current design mitigates confidentiality and integrity breaches through CP-ABE, blockchain immutability, and policy enforcement, resilience to fully adversarial fog nodes is limited. To address this, the scheme could be extended with mechanisms such as Byzantine fault-tolerant consensus protocols as in Feng et al. [15], fog node reputation systems, or Trusted Execution Environments (e.g., Intel SGX) to ensure that even untrusted or compromised fog nodes cannot undermine security. While these are beyond the current implementation scope, acknowledging them provides a pathway for adapting the framework to open and adversarial ecosystems.

6. Evaluation

This section provides a detailed comparison between the proposed framework and existing approaches, focusing on both functionality and computational efficiency. In particular, the solutions that employ Fog Computation as outsourcing mechanism were examined. Additionally, the experimental results that benchmark the performance of these related schemes are presented.

6.1. Functional analysis

Table 3 outlines a functionality comparison between the proposed scheme and existing solutions, including those presented in References [11, 13, 14, 17, 29].

From Table 3, it can be seen that all of the schemes [11, 13, 14, 17, 29] do not support the prevention of Side-Channel Attacks and Access Control Structure Optimization, which are two main features of the proposed scheme. Furthermore, all other works support the outsourced encryption and decryption process, which makes it a good point of comparison to fully differentiate the proposed scheme in its performance.

Table 3
Functional analysis

Scheme	F1	F2	F3	F4	F5
Fugkeaw et al. [11]	✓	✓	✓	X	X
Peñuelas-Angulo et al. [13]	✓	✓	✓	X	X
Ojha et al. [14]	✓	✓	✓	X	X
Apat and Sahoo [17]	✓	✓	✓	X	X
Routray et al. [29]	✓	✓	✓	X	X
Proposed Scheme	✓	✓	✓	✓	✓

Note: F1 = Outsourced Encryption; F2 = Outsourced Decryption; F3 = Fine-grained Access control; F4 = Side-Channel Attack Prevention; F5 = Access Control Structure Optimization.

6.2. Computational overhead analysis

Table 4 provides a comparison of the computational overhead associated with the proposed scheme and existing approaches. To facilitate a clear interpretation of each scheme's computational cost, the following notations are introduced.

As observed in the comparative table, the encryption and decryption processes in the proposed scheme demonstrate remarkable efficiency, particularly at both the Data Owner (DO) and Data User (DU) sides. The core operations are based entirely on constant-time XOR computations, which are significantly lighter than the cryptographic primitives used in existing schemes, such as AES, ECC, bilinear pairings, or scalar multiplications. This design choice results in a minimal and consistent computational burden, regardless of the scale of the access policy or number of attributes, offering linear complexity with minimal growth.

In contrast, other schemes such as those by Fugkeaw et al. [11] and Peñuelas-Angulo et al. [13] involve expensive operations like AES encryption, bilinear pairings, and ECC scalar multiplications, which increase computational overhead as the number of attributes or access policy nodes grows. For example, the scheme in Peñuelas-Angulo et al. [13] involves multiple group exponentiations and multiplications even at the outsourcing side, making it significantly more resource intensive. The proposed scheme, however, achieves not only computational simplicity but also reduced communication overhead for outsourcing, which is particularly beneficial for resource-constrained environments such as fog computing or IoT scenarios.

Overall, by relying on lightweight symmetric primitives and minimizing dependence on public-key operations or complex mathematical groups, the proposed scheme theoretically provides the most efficient performance profile, especially in terms of scalability and execution time.

6.3. Performance analysis

In this section, we present the results of simulations conducted to evaluate the performance of encryption and decryption operations. The experimental setup involved configuring an environment that simulates data generation and encryption, a fog layer for encryption and decryption tasks, and a blockchain component. The cryptographic operations were implemented in Rust. To assess performance, we utilized built-in timing functions provided by each programming environment to measure the execution time of individual operations in nanoseconds, which were subsequently converted to milliseconds or seconds, as appropriate.

1) Experiment environment setup

To evaluate the performance of the proposed scheme, we did experiments to measure the encryption and decryption cost of the proposed scheme and the systems in References [11, 13, 14, 17, 29]. The implementation is performed via Rust's Cryptography, and we used the Rabe Rust Library to simulate the cryptographic operations of all schemes. The experiments were executed on an Intel(R) Core(TM) i7-4790 K CPU @ 4.00 GHz and 16 GB of RAM PC simulating all written code in isolated Docker Containers in the same network, simulating a real network environment with multiple fog nodes. The parameters used in our experiments are shown in Table 5.

2) Encryption and decryption performance

In this experimental evaluation, we isolated the individual processes of Data Owner Encryption, Fog Encryption, Data User

Table 4
Computational overhead analysis

	Encryption cost		Decryption cost	
Scheme	**Data Owner**	**Outsource**	**Data User**	**Outsource**
Fugkeaw et al. [11]	AES+XOR	$(2\|T_A\|+1)$ G_0+2G_1	AES+XOR	$(2\|A_u\|+1)G_e+(2\|T_A\|+2)G_1$
Peñuelas-Angulo et al. [13]	$2\|T_A\|G_1G_m+$ $(1+\|T_A\|)G_1+$ $G_2+2G_TG_m+G_T$	$\|A_O\|G_1G_m+$ $\|T_A\|G_1+$ $\|T_A\|G_2+\|A_O\|G_TG_m$	$2G_TG_m+G_T$	$(\|A_O\|N_{da})G_1G_m+$ $\|A_u\|G_1+N_{da}G_T+$ $(1+2\|A_u\|)G_1G_e$
Ojha et al. [14]	$(2\|T_A\|+1)$ G_0+2G_1	$0.95(2\|T_A\|+1)$ G_0+2G_1	$(2\|T_A+1\|)G_0+2G_1$	$0.95(2\|T_A\|+1)G_0+2G_1$
Apat and Sahoo [17]	ECC+H	ECC	ECC	ECC
Routray et al. [29]	2ECC	$2\|T_A\|ECC$	ECC	$2\|A_u\|ECC$
Proposed Scheme	2XOR	$2((2\|T_A+1\|)G_0+2G_1)$	2XOR	$2((2\|A_u\|+1)G_e+(2\|T_A\|+2)G_1)$

Note: $|A_O|$: The number of attributes owned by data owner.

$|A_u|$: The number of attributes owned by data users.

$|T_A|$: The number of leaf nodes in access control policy.

G_0: Exponentiation and XOR operation in group G_0G_1.

G_0G_1: Exponentiation and XOR operation in group G_1G_e.

G_1G_e: Pairing operation in group G_0G_m.

G_0G_m: Multiplication operation in group G_0Z_p.

G_0Z_p: The group $\{0, 1,..., p-1\}$ multiplication modulo p.

XOR: XOR operation in 256bits data.

AES: AES encryption/decryption operation.

ECC: ECC scalar multiplication.

Table 5
Experimentation parameters

Parameter	Settings
AES Key Length	256 Bits
Policy Tree Height	1, 2, 5, 10, 20, 40, 80, 160
Data Size	40 KB

Decryption, and Fog Decryption. To ensure a fair and consistent comparison across the evaluated schemes, we varied parameters such as data size and the number of attributes within the policy tree. Computation time was specifically measured by incrementally adjusting the number of attributes in the access policy, while keeping the data size fixed at 40 KB across all test cases. The evaluation encompassed the proposed scheme, as well as the approaches in References [11, 13, 14, 17, 29]. Figures 5 and 6 illustrate the breakdown of encryption and decryption costs incurred by the Data Owner and Data User across all schemes, while Figures 7 and 8 illustrate the breakdown costs incurred by the Fog Nodes.

Figure 5 and Table 6 present the encryption cost incurred at the Data Owner (DO) side as the number of attributes increases. The proposed scheme exhibits exceptional performance, showing a flat and minimal growth in computation time due to its reliance solely

Figure 5
Encryption cost at data owner

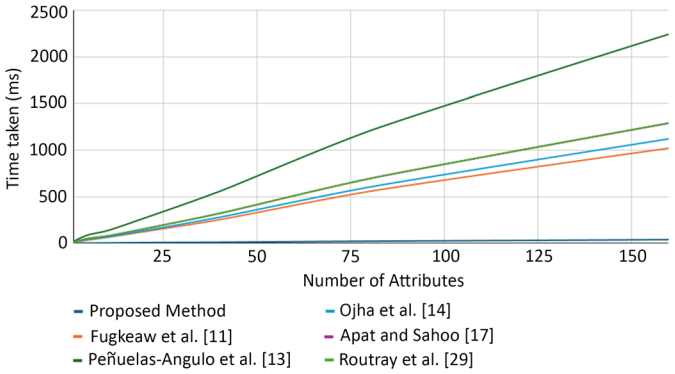


Figure 6
Decryption cost at data user

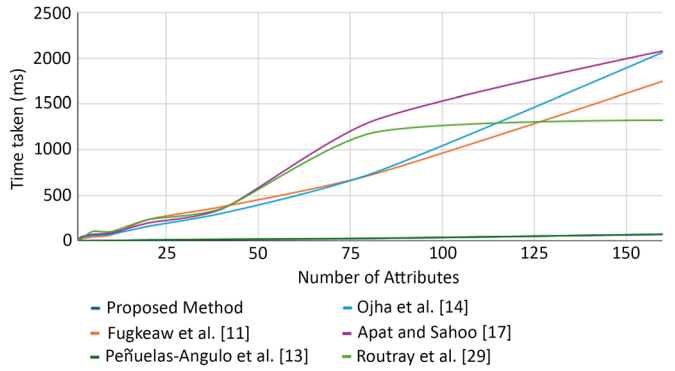


Figure 7
Encryption cost at fog node

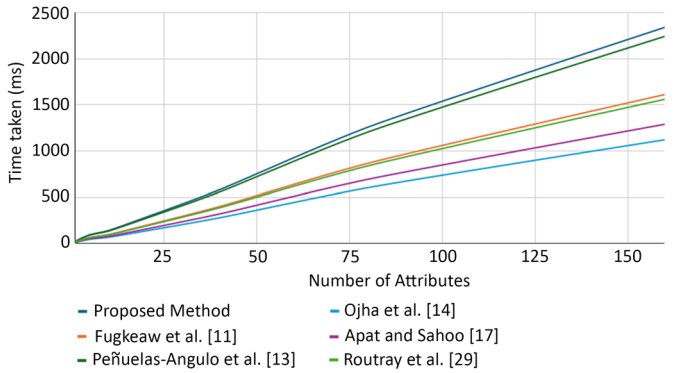
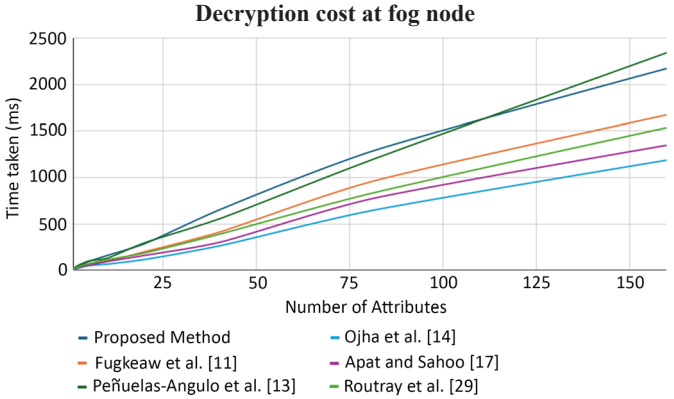


Figure 8
Decryption cost at fog node



on constant-time XOR operations. Unlike the other schemes, which incorporate cost-intensive operations such as AES encryption [11], bilinear pairing [13], or ECC and hashing [17], our method avoids such cryptographic overhead. This results in a near-constant execution time even when the number of attributes scales up to 160.

Notably, while the scheme in Fugkeaw et al. [11] also shows relatively low encryption cost, it still involves AES operations that introduce more delay compared to simple XOR, especially in constrained environments. More traditional ABE-based approaches, such as those in References [13, 14, 29], experience a sharp increase in encryption time due to the growing number of group operations and pairings required per attribute. In contrast, our scheme's performance remains largely unaffected by attribute growth, confirming its theoretical efficiency and practical scalability on the DO side.

Figure 6 and Table 7 illustrate the decryption time measured at the Data User (DU) end, again with respect to the increasing number of attributes. Here too, the proposed scheme demonstrates superior performance. The decryption time remains consistently low due to the symmetric and lightweight nature of XOR-based decryption. This lightweight requirement contrasts sharply with schemes such as in References [14, 17], which experience significant increases in decryption time due to the computational overhead introduced by ECC and pairing operations.

It is especially evident that beyond 50 attributes, most traditional schemes experience a steep rise in latency, with peaking due to its complex cryptographic constructs [17]. Meanwhile, even the relatively

Table 6
Encryption cost at data owner comparison in milliseconds (ms)

No. of attributes	Fugkeaw et al. [11]	Peñuelas-Angulo et al. [13]	Ojha et al. [14]	Apat and Sahoo [17]	Routray et al. [29]	Proposed scheme
1	8.24	15	7.5	8.625	8.64	0.64
2	17.02	40.6	20.3	23.345	23.39	1.32
5	34.25	90.5	45.25	52.04	52.14	2.09
10	63.35	134.24	67.12	77.19	77.34	3.19
20	125.04	270.4	135.2	155.48	155.79	7.23
40	252.44	556.45	278.22	319.96	320.6	10.95
80	556.46	1204.64	602.32	692.67	694.05	21.86
160	1020	2240.4	1120.2	1288.23	1290.81	38.28

Table 7
Decryption cost at data owner comparison in milliseconds (ms)

No. of attributes	Fugkeaw et al. [11]	Peñuelas-Angulo et al. [13]	Ojha et al. [14]	Apat and Sahoo [17]	Routray et al. [29]	Proposed scheme
1	10.46	0.73	7.68	10.5	11.02	1.14
2	25.09	2.52	35.23	45.66	23.78	1.81
5	44.75	2.11	60.3	72.55	103.12	3.77
10	64.09	5.02	73.25	86.46	101.55	3.92
20	232.03	12.04	158.65	195.18	232.06	9.81
40	372.62	31.37	300.8	347.39	353.05	14.52
80	715.04	75.86	724.5	1295.68	1173.08	25.73
160	1746.25	105.07	2063.53	2077.92	1322.5	74.41

efficient [11] still incurs AES-related costs that become more pronounced as the number of attributes grows. The proposed method, by avoiding asymmetric cryptography and expensive group operations altogether, ensures optimal performance for users with limited computational resources, making it particularly well-suited for fog computing and lightweight edge devices.

Figure 7 illustrates the encryption overhead incurred at the fog node (or proxy) during the outsourcing phase. While the proposed method shows increased computational time compared to some other approaches, this trade-off is by design. The scheme shifts the heavier cryptographic burden to the fog layer to minimize the load at the Data Owner (DO) side, aligning with the architectural goal of offloading complex operations away from client devices. Compared to the schemes in Ojha et al. [14] as well as Apat and Sahoo [17], which show relatively lower fog-side overhead, our approach incorporates additional functionalities, such as encrypted attribute mapping and electing quorums, which contribute to the slightly elevated execution time.

Despite this, it is important to note that the fog infrastructure is typically more capable than end-user devices, and thus more suitable for handling complex tasks like key transformation and access structure evaluation. Therefore, the increase in encryption time at the fog layer does not compromise the practicality of the scheme, especially in distributed systems with abundant edge computing resources.

As shown in Figure 8, the proposed scheme also incurs moderately higher decryption costs at the fog layer when compared to alternatives. This is primarily attributed to the additional responsibilities placed on the fog, including quorum election and on-chain data re-encryption required to enforce dynamic access control. These operations, while adding complexity, are crucial for supporting flexible and secure access policies in decentralized environments.

Compared to the schemes in References [11, 13, 29], which perform relatively faster at the fog level, our method emphasizes security guarantees and resistance to unauthorized reconstruction, even at the cost of added delay. The trade-off reflects a deliberate shift in system design: reducing the burden at DO and DU while centralizing computationally intensive tasks in the fog layer, where resources are typically more robust and scalable.

In conclusion, the experimental results validate that the proposed scheme achieves the lowest computational overhead at both the Data Owner and Data User ends for encryption and decryption operations. These results are consistently observed across varying numbers of attributes, confirming the scalability and lightweight nature of the XOR-based design. Although the performance at the fog layer is not the most optimal due to added cryptographic responsibilities and blockchain-related tasks, this is a justified trade-off that aligns with the goal of reducing client-side burden.

Overall, the scheme offers a well-balanced architecture that emphasizes user efficiency without compromising on security or access control flexibility, making it particularly suitable for fog-assisted blockchain environments.

7. Discussion

The results of the experimental evaluation highlight a fundamental strength of the proposed scheme: its ability to deliver lightweight,

scalable access control mechanisms in high-throughput environments without compromising security. This is particularly critical in financial and transactional systems where performance bottlenecks or cryptographic inefficiencies can translate into operational delays or security vulnerabilities. The use of a dual-layer encryption design, employing constant-time XOR operations at both the Data Owner (DO) and Data User (DU) sides, substantially reduces computational overhead, as confirmed by both theoretical and empirical analyses. In contrast to many prior approaches that burden edge devices with asymmetric or pairing-based cryptographic operations, our scheme distributes complexity more intelligently across system layers.

A key architectural feature that supports this balance is the strategic use of fog nodes. Although these nodes exhibit moderately higher encryption and decryption overhead, they are deliberately tasked with handling the more resource-intensive components. This reflects an intentional shift in system design, prioritizing user-side efficiency while leveraging fog-level compute capacity for complex cryptographic functions. While the blockchain layer, underpinned by a Proof of Authority (PoA) consensus mechanism, serves as both a secure ledger for encrypted transactions and a trusted platform for enforcing access control via smart contracts.

Beyond performance, the proposed system is equipped with multiple layers of defense against a range of known attacks. These include Sybil Attacks, Double Voting/Spending, Denial-of-Service (DoS), Man-in-the-Middle (MitM), Replay Attacks, and Side-Channel Attacks.

While the proposed method has proven highly effective at reducing user-end overhead, it also reveals opportunities for further refinement, especially within the fog layer. As noted in Figures 7 and 8, encryption and decryption at the fog level experience higher latency due to quorum coordination and smart contract-triggered re-encryption. These are areas where future research can focus on improving process efficiency, perhaps by leveraging parallelism or adaptive quorum sizes.

In summary, this discussion confirms that the proposed scheme not only addresses long-standing challenges in secure access control for blockchain-based systems but also delivers tangible performance and security advantages through thoughtful architectural choices. Its layered, role-based design ensures a high degree of adaptability, making it well-suited for deployment in real-world, security-sensitive applications such as finance, healthcare, and government systems.

8. Conclusion

The implementation of the proposed scheme addresses key limitations in existing access control solutions for financial transactions and other high-volume, high-throughput environments. Many current approaches are not optimized for the performance demands of such systems, often relying on computationally intensive encryption mechanisms that overwhelm end-user devices. Furthermore, these methods typically lack robust protection against advanced threats like side-channel attacks. The scheme overcomes these issues by introducing a two-layer lightweight encryption scheme tailored for fine-grained, scalable access control. The architecture incorporates a randomized sharding algorithm for dynamic fog node selection and a novel quorum election mechanism, enabling efficient offloading of cryptographic operations. Additionally, the use of multilayer encryption at both the transaction and block level enhances resistance to side-channel threats and ensures stronger data confidentiality.

Theoretical analysis and experimental results confirm the practicality and efficiency of the proposed system, particularly in reducing computational overhead at the DO and DU sides. These benefits are especially significant in financial systems, where security and performance must be tightly balanced. However, the fog layer remains an area for improvement, with observed performance bottlenecks caused by coordination complexity across nodes. In addition, the current implementation is evaluated primarily under controlled, consortium-based conditions. While this reflects realistic deployment in permissioned environments such as finance, healthcare, or enterprise settings, the scheme does not yet fully address open or adversarial environments where fog nodes cannot be assumed trustworthy. Similarly, while the adversary model accounts for semi-honest fog nodes and colluding users, resilience against stronger adversaries remains a limitation that warrants further exploration.

Future work will therefore focus on optimizing fog-level processes to improve scalability. Additional directions include advancing key management, especially revocation in CP-ABE and performing production-scale evaluations in realistic environments and strengthening the native integration between blockchain and fog computing. By directly addressing the core limitations of existing solutions, the scheme provides a comprehensive, efficient, and secure framework for privacy-preserving data access in decentralized systems.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in GitHub at https://github.com/Fraunhofer-AISEC/rabe.

Author Contribution Statement

Leon Wirz: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Pattarasinee Bhattarakosol:** Validation, Writing – review & editing, Supervision, Project administration.

References

[1] Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2020). A survey on the security of blockchain systems. *Future Generation Computer Systems*, *107*, 841–853. https://doi.org/10.1016/j.future.2017.08.020

[2] Wen, B., Wang, Y., Ding, Y., Zheng, H., Qin, B., & Yang, C. (2023). Security and privacy protection technologies in securing blockchain applications. *Information Sciences*, *645*, 119322. https://doi.org/10.1016/j.ins.2023.119322

[3] Jia, P., Zhang, J., Zhao, B., Li, H., & Liu, X. (2023). Privacy-preserving association rule mining via multi-key fully homomorphic encryption. *Journal of King Saud University - Computer and Information Sciences*, *35*(2), 641–650. https://doi.org/10.1016/j.jksuci.2023.01.007

[4] Keshta, I., Aoudni, Y., Sandhu, M., Singh, A., Xalikovich, P. A., Rizwan, A., ..., & Lalar, S. (2023). Blockchain aware proxy re-encryption algorithm-based data sharing scheme. *Physical Communication*, *58*, 102048. https://doi.org/10.1016/j.phycom.2023.102048

[5] Huang, T. (2023). Transaction database encryption technology based on blockchain technology. In *2023 8th International*

Conference on Information Systems Engineering, 342–345. https://doi.org/10.1109/ICISE60366.2023.00078

[6] Jadhav, H., & Chandre, P. (2016). Association rule mining methods for applying encryption techniques in transaction dataset. In *2016 International Conference on Computer Communication and Informatics*, 1–5. https://doi.org/10.1109/ICCCI.2016.7479939

[7] Bethencourt, J., Sahai, A., & Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, 321–334. https://doi.org/10.1109/SP.2007.11

[8] Zhang, J.-X., & Zhang, L.-Y. (2017). Anonymous CP-ABE against side-channel attacks in cloud computing. *Journal of Information Science and Engineering*, *33*(3), 789–805. https://doi.org/10.6688/JISE.2017.33.3.12

[9] Sarma, R., Kumar, C., & Barbhuiya, F. A. (2022). MACFI: A multi-authority access control scheme with efficient ciphertext and secret key size for fog-enhanced IoT. *Journal of Systems Architecture*, *123*, 102347. https://doi.org/10.1016/j.sysarc.2021.102347

[10] Fugkeaw, S., Wirz, L., & Hak, L. (2023). An efficient medical records access control with auditable outsourced encryption and decryption. In *2023 15th International Conference on Knowledge and Smart Technology*, 1–6. https://doi.org/10.1109/KST57286.2023.10086904

[11] Fugkeaw, S., Wirz, L., & Hak, L. (2023). Secure and lightweight blockchain-enabled access control for fog-assisted IoT cloud based electronic medical records sharing. *IEEE Access*, *11*, 62998–63012. https://doi.org/10.1109/ACCESS.2023.3288332

[12] Shruti, Rani, S., Shabaz, M., Dutta, A. K., & Ahmed, E. A. (2024). Enhancing privacy and security in IoT-based smart grid system using encryption-based fog computing. *Alexandria Engineering Journal*, *102*, 66–74. https://doi.org/10.1016/j.aej.2024.05.085

[13] Peñuelas-Angulo, A., Feregrino-Uribe, C., & Morales-Sandoval, M. (2024). A revocable multi-authority attribute-based encryption scheme for fog-enabled IoT. *Journal of Systems Architecture*, *155*, 103265. https://doi.org/10.1016/j.sysarc.2024.103265

[14] Ojha, S., Paygude, P., Dhumane, A., Rathi, S., Bidve, V., Kumar, A., & Devale, P. (2024). A method to enhance privacy preservation in cloud storage through a three-layer scheme for computational intelligence in fog computing. *MethodsX*, *13*, 103053. https://doi.org/10.1016/j.mex.2024.103053

[15] Feng, P., Ma, Y., Li, B., Han, B., & Fang, T. (2024). Trustworthy fog: A reputation-based consensus method for IoT with blockchain and fog computing. *Computers and Electrical Engineering*, *120*, 109749. https://doi.org/10.1016/j.compeleceng.2024.109749

[16] dos Santos, F. C., Duarte-Figueiredo, F., de Grande, R. E., & dos Santos, A. L. (2024). Enhancing a fog-oriented IoT authentication and encryption platform through deep learning-based attack detection. *Internet of Things*, *27*, 101310. https://doi.org/10.1016/j.iot.2024.101310

[17] Apat, H. K., & Sahoo, B. (2024). A blockchain assisted fog computing for secure distributed storage system for IoT applications. *Journal of Industrial Information Integration*, *42*, 100739. https://doi.org/10.1016/j.jii.2024.100739

[18] Zhang, J., Zhang, W., Wei, X., & Liu, H. (2024). EPri-MDAS: An efficient privacy-preserving multiple data aggregation scheme without trusted authority for fog-based smart grid. *High-Confidence Computing*, *4*(4), 100226. https://doi.org/10.1016/j.hcc.2024.100226

[19] Sarma, R., & Moulik, S. (2024). e-SAFE: A secure and efficient access control scheme with attribute convergence and user revocation in fog enhanced IoT for E-Health. *Journal of Information Security and Applications*, *85*, 103859. https://doi.org/10.1016/j.jisa.2024.103859

[20] Tao, Y., Zhu, Y., Ge, C., Zhou, L., Zhou, S., Zhang, Y., ..., & Fang, L. (2024). ORR-CP-ABE: A secure and efficient outsourced attribute-based encryption scheme with decryption results reuse. *Future Generation Computer Systems*, *161*, 559–571. https://doi.org/10.1016/j.future.2024.07.040

[21] Shen, X., Zhu, L., Xu, C., Sharif, K., & Lu, R. (2020). A privacy-preserving data aggregation scheme for dynamic groups in fog computing. *Information Sciences*, *514*, 118–130. https://doi.org/10.1016/j.ins.2019.12.007

[22] Qin, X., Huang, Y., Yang, Z., & Li, X. (2021). LBAC: A lightweight blockchain-based access control scheme for the internet of things. *Information Sciences*, *554*, 222–235. https://doi.org/10.1016/j.ins.2020.12.035

[23] Hou, X., Zhang, L., Wu, Q., & Rezaeibagha, F. (2023). Collusion-resistant dynamic privacy-preserving attribute-access control scheme based on blockchain. *Journal of King Saud University - Computer and Information Sciences*, *35*(8), 101658. https://doi.org/10.1016/j.jksuci.2023.101658

[24] Mu, T., Lai, Y., Feng, G., Lyu, H., Yang, H., & Deng, J. (2023). A user-friendly attribute-based data access control scheme for smart grids. *Alexandria Engineering Journal*, *67*, 209–217. https://doi.org/10.1016/j.aej.2022.12.041

[25] Deng, X., Peng, C., Yang, H., Peng, Z., & Zhong, C. (2024). A dynamic data access control scheme for hierarchical structures in big data. *Computer Communications*, *220*, 128–137. https://doi.org/10.1016/j.comcom.2024.04.006

[26] Hundera, N. W., Aftab, M. U., Mesfin, D., Dioubi, F., Xu, H., & Zhu, X. (2024). An efficient heterogeneous online/offline anonymous certificateless signcryption with proxy re-encryption for Internet of Vehicles. *Vehicular Communications*, *49*, 100811. https://doi.org/10.1016/j.vehcom.2024.100811

[27] Habibi, P., Farhoudi, M., Kazemian, S., Khorsandi, S., & Leon-Garcia, A. (2020). Fog computing: A comprehensive architectural survey. *IEEE Access*, *8*, 69105–69133. https://doi.org/10.1109/ACCESS.2020.2983253

[28] Dong, Y., Li, Y., Cheng, Y., & Yu, D. (2024). Redactable consortium blockchain with access control: Leveraging chameleon hash and multi-authority attribute-based encryption. *High-Confidence Computing*, *4*(1), 100168. https://doi.org/10.1016/j.hcc.2023.100168

[29] Routray, K., Bera, P., & Mishra, S. (2025). Efficient and secure fine-grained authorization for fog-enabled cyber-physical systems: Towards trust and autonomy in digital sovereignty. *Procedia Computer Science*, *254*, 240–249. https://doi.org/10.1016/j.procs.2025.02.083

[30] Narla, S., Peddi, S., Valivarthi, D. T., Kethu, S. S., Natarajan, D. R., & Kurniadi, D. (2025). FOG computing based energy efficient and secured iot data sharing using SGSOA and GMCC. *Sustainable Computing: Informatics and Systems*, *46*, 101109. https://doi.org/10.1016/j.suscom.2025.101109

[31] Yang, X., Luo, X., Liao, Z., Wang, W., Du, X., & Li, S. (2025). A CP-ABE-based access control scheme with cryptographic reverse

firewall for IoV. *Journal of Systems Architecture*, *160*, 103331. https://doi.org/10.1016/j.sysarc.2025.103331

[32] Irving, R. W., & Love, L. (2003). The suffix binary search tree and suffix AVL tree. *Journal of Discrete Algorithms*, *1*(5–6), 387–408. https://doi.org/10.1016/S1570-8667(03)00034-0

[33] Deekumpa, P., & Sooraksa, P. (2014). Associate rule minimization using Boolean algebra set function. In *The 4th Joint International Conference on Information and Communication Technology, Electronic and Electrical Engineering*, 1–5. https://doi.org/10.1109/JICTEE.2014.6804099

How to Cite: Wirz, L., & Bhattarakosol, P. (2026). Sharded Fog Architecture for Scalable CP-ABE with Minimized Policy Trees in Blockchain Systems. *Journal of Computational and Cognitive Engineering*, *5*(2), 304–320. https://doi.org/10.47852/bonviewJCCE52026785