EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Privacy-Preserving Intelligent Financial Information Processing in Cloud-Edge Distributed Systems Based on Homomorphic Encryption and MapReduce

Cuilan Wang[1,*] and Linghui Zhao[1]

[1]School of Economics and Management, Anyang Vocational and Technical College, Anyang Henan 455000, China

Abstract

Privacy-preserving financial big-data processing must simultaneously control plaintext exposure and the computational cost of distributed analytics. This paper proposes a cloud-edge MapReduce-ECC framework in which edge terminals perform local validation, sharding, feature fusion, and ciphertext generation, while cloud Reduce nodes aggregate only matched protected shards and HDFS stores encrypted blocks with rack-aware replication. The design clarifies key separation, ciphertext matching, fault recovery, and compliance-oriented data-localization rules. Experiments on a 5.5-TB operational e-commerce financial dataset show that 42 Map nodes provide the best observed parallel-processing trade-off. The HDFS deployment maintains storage-task completion and post-storage availability close to 100%, while the stated trust-boundary design prevents plaintext financial shards from being transmitted to ordinary cloud workers. These results indicate that the framework provides a scalable privacy-oriented processing pipeline for distributed financial information systems.

Keywords: distributed networks, financial big data, privacy protection, homomorphic encryption, elliptic curve cryptography, MapReduce, HDFS, cloud-edge collaboration

Received on 24 June 2026, accepted on 14 October 2026, published on 13 November 2026

Copyright © 2026 Cuilan Wang *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.13699

1. Introduction

Data-intensive financial management increasingly includes corporate risk early-warning, big-data decision support, cloud-based branch financial systems, and cloud-enabled accounting analytics [1-4]. These applications operate on large, heterogeneous, and frequently updated records and therefore require both scalable computation and stronger privacy controls. Conventional centralized processing or isolated analytics pipelines can become bottlenecks as data volume and cross-system integration grow, particularly when sensitive financial information must be processed across distributed infrastructure.

Existing studies address these requirements through several partially separated directions. Harmony-search-assisted K-Medoids improves clustering performance in general data-mining applications, but it is not a privacy-preserving financial-computing mechanism [5]. Kang and Wu combine blockchain, federated learning, and local differential privacy in distributed learning [6]; Li et al. design blockchain-based secure storage and access control for supply-chain finance [7]; and Telepathy optimizes NVRAM/RDMA distributed storage rather than confidentiality [8]. In EAI Endorsed Transactions on Scalable Information Systems, Gupta and Dwivedi proposed a Fortified Secured MapReduce layer positioned

*Corresponding author. Email: 13837231361@163.com

between HDFS and MapReduce to strengthen privacy and security in big-data processing [9], while Tian developed a privacy-protection and security-management mechanism for distributed financial systems using privacy-preserving data generation, blockchain-based access control, and attack detection [10]. Recent finance-oriented homomorphic-encryption research includes cloud credit-risk prediction [11], digital-finance regulatory verification [12], central-bank reserve protection [13], federated financial-data management [14], private financial reporting [15], and privacy-preserving federated credit-risk modeling [16]. Compute-efficient Byzantine-robust federated learning with fully homomorphic encryption demonstrates the broader feasibility of scalable ciphertext computation [17], while modified homomorphic encryption combined with federated learning has also been studied in blockchain-enabled IoT outside the financial domain [18]. Nevertheless, these studies typically optimize encrypted analytics, collaborative learning, blockchain governance, or storage/communication separately. They do not provide the same end-to-end coupling used here: MapReduce sharding determines independent financial-data processing units; Map workers perform feature fusion and ECC-based protection per shard; Reduce workers aggregate only compatible protected outputs; and HDFS stores the resulting encrypted blocks with rack-aware fault tolerance.

Table 1. Comparison with representative privacy-preserving distributed financial computing studies

Study	Core technique	Distributed setting	Main limitation relative to this work
Kang and Wu [6]	Blockchain + federated learning + differential privacy	Distributed clients	No MapReduce-ECC shard/ciphertext coupling
Li et al. [7]	Blockchain secure storage and access control	Supply-chain finance ledger	Ledger-focused; no ciphertext parallel analytics
Naresh and Ayyappa [11]	Paillier HE + DNN	Cloud credit-risk prediction	Encrypted inference/training, not Hadoop sharding
Ren et al. [12]	HE + blockchain	Digital-finance regulation	Regulatory verification rather than scalable MapReduce processing
Shafin and Reno [13]	Blockchain + FHE + AWS	Central-bank reserve management	Higher architectural complexity; not HDFS/MapReduce oriented
Guo [14]	HE + MPC + federated learning	Distributed financial data management	Collaborative training focus; no ciphertext HDFS pipeline
Alaklabi [15]	Blockchain + FL + DP + HE	Private financial reporting	Governance/FL focus; no shard-level HDFS fault tolerance
Naresh and Ayyappa [16]	HE/DP federated credit-risk model	Five distributed financial clients	Model-update protection rather than full data lifecycle
Gupta and Dwivedi [9]	Fortified Secured MapReduce privacy/security layer	HDFS-MapReduce big-data platform	General big-data privacy; no finance-specific EC-ElGamal aggregation or cloud-edge task partitioning
Tian [10]	Privacy-preserving data generation + blockchain access control + attack detection	Distributed financial systems	Security-management focus; no MapReduce/HDFS protected-aggregation pipeline
This work	MapReduce + RF fusion + ECC-based ciphertext aggregation + HDFS	Cloud-edge financial big data	Joint sharding, ciphertext processing, storage, and recovery

The contributions are therefore threefold. (1) A cloud-edge task-partitioning mechanism couples edge-local shard preparation with cloud-side MapReduce execution so that raw financial records do not need to be exposed to cloud aggregation nodes. (2) An ECC-based protected aggregation workflow adds explicit key separation, ciphertext matching rules, Reduce-node aggregation steps, and authorized decryption verification, improving the reproducibility of the security pipeline. (3) Encrypted HDFS storage is extended with rack-aware re-replication and consistency checks, while the evaluation protocol unifies latency, encryption overhead, parallel throughput, storage availability, and plaintext-exposure analysis.

Cloud computing provides elastic computation and storage for large-scale information systems [19], while data-protection-aware offloading and finance-oriented edge platforms move privacy-sensitive processing closer to data sources [20,21]. Hadoop provides a distributed processing and storage stack in which MapReduce performs parallel task execution and the Hadoop Distributed File System (HDFS) supplies scalable replicated storage [22,23]. Homomorphic encryption enables selected operations to be executed without exposing plaintext; foundational work on fully homomorphic encryption, Paillier encryption, and ElGamal encryption establishes the cryptographic basis for protected computation [24-26]. Elliptic Curve Cryptography (ECC) provides compact public-key operations based on the elliptic-curve discrete logarithm problem [27,28], and EC-ElGamal variants have been explicitly studied for additive homomorphic operations over elliptic-curve-encoded data [29]. Edge-computing surveys further characterize the latency and communication motivations for near-data processing [30,31], while heterogeneous-cloud scheduling research highlights the practical effect of cost and resource variability on distributed big-data execution [32]. Based on these components, this study targets the full financial-data lifecycle rather than a single encryption or storage stage: edge-local preparation, parallel shard fusion, protected aggregation, secure cloud-edge transmission, and distributed encrypted storage.

2. Intelligent Processing Methods for Financial Information in a Big Data Environment

This study uses the Hadoop open-source distributed computing platform to implement financial-data sharding, parallel fusion, protected aggregation, and encrypted distributed storage. Hadoop MapReduce provides the parallel execution model, whereas the Hadoop Distributed File System (HDFS) provides replicated storage [22,23]. Related work in EAI Endorsed Transactions on Scalable Information Systems has also introduced an explicit privacy/security layer between HDFS and MapReduce [9], supporting the need to integrate protection mechanisms directly with the distributed processing stack. In the cloud-edge deployment, edge financial terminals retain raw records, perform local schema validation and shard preparation, and generate protected shard outputs. Cloud Map and Reduce tasks execute scalable fusion and ciphertext aggregation, and only encrypted blocks are persisted in HDFS. The resulting architecture is shown in Figure 1.

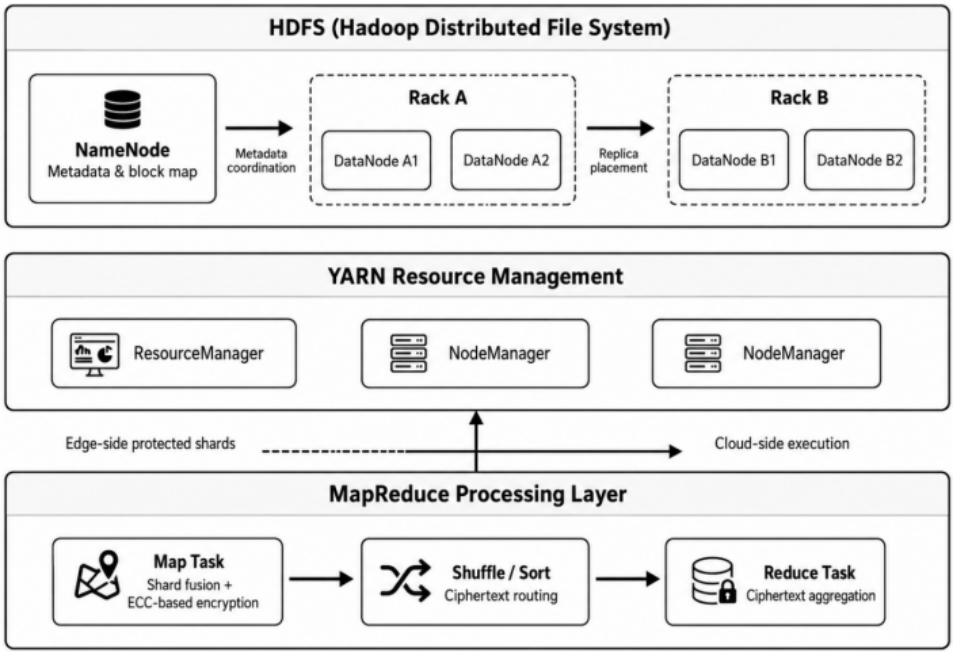


Figure 1. Architecture of hadoop cloud computing platform

As shown in Figure 1, the NameNode maintains file-system namespace and block-location metadata, whereas DataNodes store encrypted HDFS blocks. At the processing layer, Map Tasks receive prepared financial shards, perform feature fusion and ECC-based encoding, Shuffle/Sort routes ciphertext records by aggregation key, and Reduce Tasks combine only records with compatible metadata. YARN coordinates resource allocation between these tasks. In the proposed cloud-edge workflow, plaintext financial records remain at trusted edge terminals; protected shard outputs cross the edge-cloud boundary, and the cloud performs aggregation and storage without requiring access to raw records.

2.1. Financial Information Big Data Sharding Integration and Encryption Processing Based on the MapReduce Cloud Computing Model

The MapReduce model contains partition/shard generation, Map processing, Shuffle/Sort routing, and Reduce aggregation [22]. The partitioning function divides the original multi-source financial collection into independently addressable shards. Map Tasks perform random-forest (RF) feature fusion and ECC-based protection on each shard, Shuffle/Sort groups outputs with the same aggregation metadata, and Reduce Tasks aggregate compatible ciphertext outputs. This mapping makes each security operation traceable to a native distributed-computing component and supports horizontal scaling by increasing the number of Map workers.The structural diagram of the MapReduce cloud computing model for financial information big-data fusion and encryption processing is shown in Figure 2.

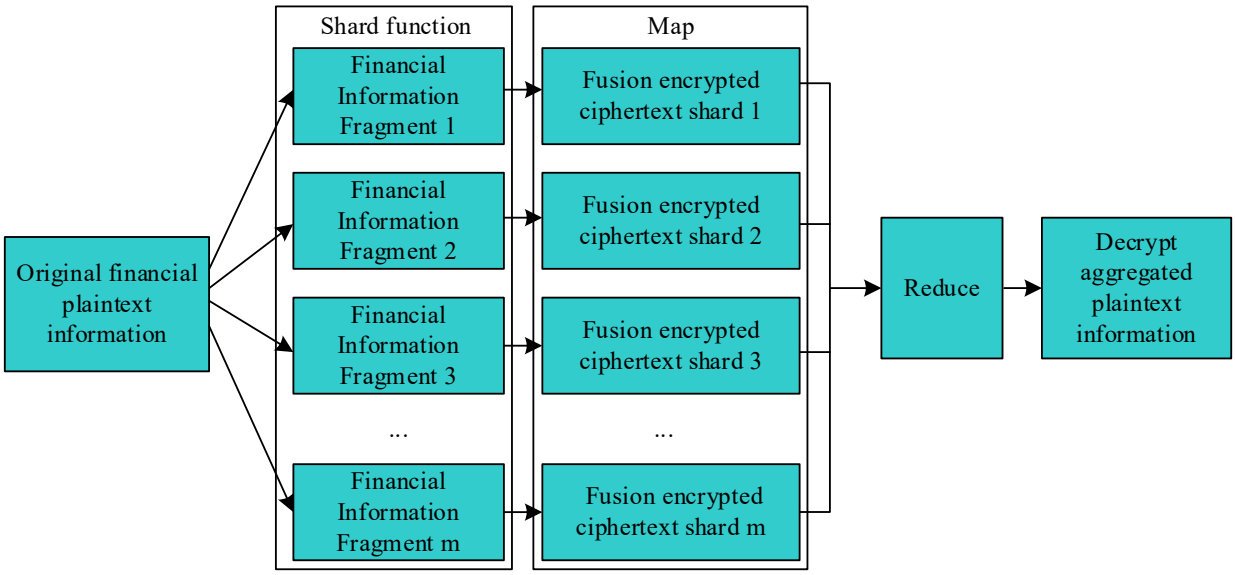


Figure 2. Structural diagram of the MapReduce cloud computing model for financial information big data fusion and encryption processing

2.1.1 Parallel Fusion Processing of Big Data Financial Information

Let $A = \{f(a_1), f(a_2), \cdots, f(a_m)\}$ represent a massive collection of multi-source heterogeneous raw financial information, where $a_1, \cdots, a_m$ denotes various different subsets of financial information, m indicates the number of subsets in this financial information collection, and f represents the mapping inference function of the information collection.

Due to the large and complex data contained in the financial information sample collection, directly implementing fusion processing is quite challenging. Therefore, the collection A is first placed in the MapReduce cloud computing model, where its sharding function divides the collection A into m financial information subset shards $a_1, \cdots, a_m$. In conjunction with the Map nodes in this model, several decision trees within the random forest algorithm are called to perform parallel fusion processing on each financial information subset shard $a_1, \cdots, a_m$, reducing the generalization error of big data financial information while achieving a rough classification of the original financial information set A. Among them, the sharding process of the original massive financial

information collection A in the MapReduce cloud computing model is as follows: the original massive financial information collection A is divided into m financial information subset shards $a_1, \cdots, a_m$, where the size l_{ai} of the i th subset shard a_i is:

$$l_{a_i} = \begin{cases} \left[\left(L/\hat{\kappa}\right)/m\right] \times \kappa, & i < m \\ L - (m-1) \times \left[\left(L/\hat{\kappa}\right)/m\right] \times \kappa, & i = m \end{cases} \quad (1)$$

Equation (1) determines the size of each financial-data shard, so it directly controls Map-task granularity and the amount of plaintext assigned to each parallel worker.

In the formula, κ represents the expansion parameter for the number of plaintext encrypted bits of the financial information subset; L indicates the size of the original massive financial information collection A. Let pos represent the position of the financial information being processed in the original financial information collection A, used to determine whether this financial information subset shard has finished reading; filebuffer is used to store the information of each financial information subset shard $a_1, \cdots, a_m$, with subset shard a_i stored using key, and the value being the filebuffer that saves the i th subset shard a_i. After completing the sharding of the original financial information collection A, each financial information subset shard $a_1, \cdots, a_m$ is allocated to various Map nodes for subsequent parallel fusion processing.

By invoking the j th decision tree within the random forest algorithm at the j th layer Map node to process the i th subset shard a_i of the original financial information set A, the decision tree d_j at this layer Map node can be represented as:

$$d_j = d(a_i, \omega_j) \quad (2)$$

Equation (2) defines the output of the j-th random-forest decision tree for shard a_i under its random parameter vector, providing a reproducible per-shard fusion primitive.

In the formula, ω_j represents the parameters used to capture random factors in the decision tree. Based on each decision tree d_j, the decision tree ensemble multi-source heterogeneous financial information fusion model $D$is constructed as follows:

$$D = \{d_1, d_2, \cdots, d_j, \cdots, d_m\} \quad (3)$$

Equation (3) collects the individual decision trees into the ensemble model D used for parallel feature fusion across Map workers.

If the classification evaluation probability of different types of financial information in the i th financial information subset slice is $\mathrm{ep}[p|a_i]$, then the information classification evaluation probability value of the entire financial information set can be expressed as:

$$\mathrm{ep}[p|A] = \frac{1}{m}\sum_{i=1}^{m} \mathrm{ep}[p|a_i] \quad (4)$$

Equation (4) averages shard-level class-evaluation probabilities to obtain the collection-level evaluation probability, thereby linking local Map results to the global fusion output.

In the process of fusing massive raw financial information, if the trend of classification evaluation is correct, then $\varsigma(a_i, p) > 0$, where $\varsigma(a_i, p)$ represents the boundary of financial information fusion preprocessing. At this time, the generalization error value μ of the financial information fusion preprocessing using the random forest algorithm satisfies the following conditions:

$$\mu = \delta[m(a_i, p)] < 0 \quad (5)$$

Equation (5) defines the generalization-error condition used to judge whether the fused classification remains within the specified preprocessing boundary.

In the formula, δ represents the expected classification evaluation during the fusion process of massive raw financial information. Financial information fusion preprocessing can fully utilize the advantages of the MapReduce cloud computing model in overall computation, implementing coarse classification on massive raw multi-source heterogeneous financial information sets in a big data environment, extracting attribute features between different categories of financial information, significantly reducing the dimensionality of financial information, and helping to enhance and improve the classification performance and accuracy of financial information.

2.1.2 Parallel Encryption Processing of Financial Information Big Data Slices

Because financial information is highly sensitive, the protection layer uses Homomorphic Encryption (HE) concepts together with Elliptic Curve Cryptography (ECC)-based encoding before protected shards leave the trusted processing boundary [24-29]. Therefore, after the parallel fusion processing of each information subset slice a_i in the aforementioned massive raw financial information collection A, each Map node continues to implement parallel encryption processing on the fused financial information subset slices a_i, using ECC-based protected encoding before cloud-side aggregation. The specific protection process is as follows:

(i) Homomorphic Encryption of Financial Information

Homomorphic Encryption (HE) permits selected algebraic operations on protected values without revealing the underlying plaintext. Finance-oriented applications include credit-risk prediction, regulatory verification, reserve protection, financial-data management/reporting, and privacy-preserving federated credit-risk modeling [11-16], while compute-efficient FHE in distributed learning and foundational HE/EC-ElGamal studies provide broader cryptographic and systems support [17,24-29]. In this

paper, Equation (6) is an application-level operation-preserving requirement. Let the encryption transformation after fusion be CK, the corresponding decryption transformation be DK, and the plaintext financial shard values be $a_i^{'}$ and $a_j^{'}$; the protected workflow satisfies the following consistency properties:

$$\begin{cases} \mathrm{DK}\left[\mathrm{CK}\left(a_i^{'}\right) * \mathrm{CK}\left(a_{j_i}^{'}\right)\right] = a_i^{'} * a_j^{'} \\ \mathrm{DK}\left[\mathrm{CK}\left(a_i^{'}\right) + \mathrm{CK}\left(a_{j_i}^{'}\right)\right] = a_i^{'} + a_j^{'} \end{cases} \quad (6)$$

Equation (6) should be interpreted as an application-level homomorphic consistency requirement rather than a claim that standard ECC alone is a general-purpose fully homomorphic encryption scheme. In the implemented workflow, ECC supplies a compact group-based ciphertext representation, and only the encoded aggregation operations explicitly defined in Equations (10)-(17) are supported. Arbitrary ciphertext circuits are outside the scope of this paper. This clarification separates elliptic-curve group operations from unrestricted plaintext arithmetic and avoids conflating ECC with fully homomorphic encryption [24-29].

(ii) Financial Information Elliptic Curve Encryption

Elliptic Curve Cryptography (ECC) is based on the intractability of the elliptic curve discrete logarithm problem. In the encryption of financial information, the fused financial plaintext information fragment $a_i^{'}$ can be embedded into random points on an elliptic curve. During decryption, the points are decoded back into the plaintext information. Through the public-key and private-key system, the security of financial information during transmission and storage is ensured.The security of elliptic curve cryptography is based on the difficulty of solving the discrete logarithm problem on elliptic curves. Let Bbe an elliptic curve and B_1 and B_2 be two random points on the curve, where $B_2 = xB_1$. In the encryption of plaintext financial information, B_2 can represent the public key, B_1 represents the base point for elliptic curve cryptography operations, and x can represent the private key. Solving for x is the elliptic curve discrete logarithm problem. Elements x, y, etc., in the base field C participate in the construction of the elliptic curve equation. The elements in the base field C satisfy the following conditions:

$$y^2 + e_1xy + e_2y = x^3 + e_3x^2 + e_4x + e_5 \quad (7)$$

In the equation, e_1, e_2 and x, y all represent elements in the finite field Cq, where q denotes a prime number greater than 3; e_3 and e_4 represent the coefficients of the x^2 term and the x term, respectively; and e_5 represents the constant term in the equation. By coordinate transformation, equation (7) can be converted to $B: y^2 = x^3 + e_1x + e_2$. For two points $\dot{Q}(x_1, y_1)$ and $B_2(x_2, y_2)$ on the elliptic curve, where $\dot{Q}(x_1, y_1)$ is the random point on the elliptic curve where the financial plaintext information fragment a_i', after fusion processing, is embedded, and the slope of the line $\dot{Q}B$ connecting these two points is $\Delta = (y_2 - y_1)/(x_2 - x_1)$. The line $\dot{Q}B$ intersects the elliptic curve at another point $G(x_3, y_3)$, and this intersection point G is the negative element of the sum of B_2 and $\dot{Q}$, i.e., $\dot{Q} + B_2 = -G$. Where:

$$\begin{cases} x_3 = \Delta^2 - x_1 - x_2 \\ y_3 = \Delta x_1 - \Delta x_3 - x_1 \end{cases} \quad (8)$$

The distribution of points and curves on the elliptic curve is shown in Figure 3.

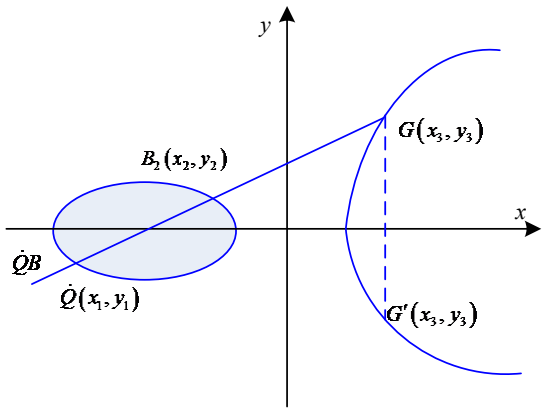


Figure 3. Distribution of points and curves on an elliptic curve

To find the double of the operation point $\dot{Q}$, draw a tangent line through this point and find another intersection point $G'(x_3, y_3)$, then $\dot{Q} + \dot{Q} = 2\dot{Q} = -G'$, where:

$$\begin{cases} x_3 = [(3x_1^2 + e_1)/2y_1]^2 - 2x_1 \\ y_3 = [(3x_1^2 + e_1)/2y_1](x_1 - x_3) - y_1 \end{cases} \quad (9)$$

When encrypting financial plaintext information fragments a_i' using elliptic curves, the plaintext needs to be embedded at a random point on the elliptic curve. During decryption, the point on the elliptic curve must be decoded back to the financial plaintext information. The encryption and decryption are achieved through the public key B_2 and private key x. The process of encrypting financial information using elliptic curves is illustrated in Figure 4.

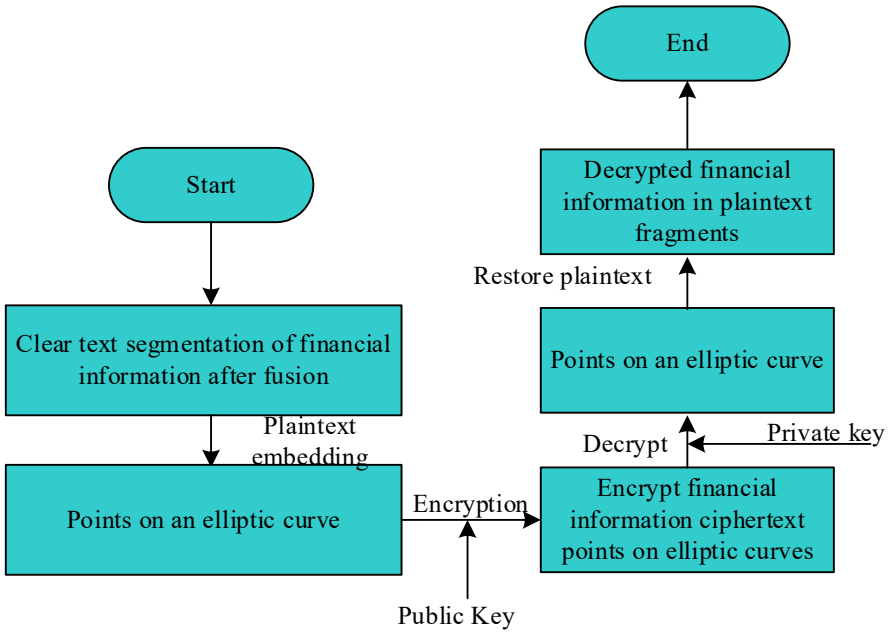


Figure 4. Diagram illustrating the financial information encryption process using an elliptic curve

(iii) ECC-Based Protected Aggregation of Financial Information

This subsection describes ECC-based protected aggregation rather than unrestricted homomorphic multiplication. Each fused financial shard is encoded as an elliptic-curve point, randomized, and transformed into a ciphertext pair. EC-ElGamal-type constructions support additive homomorphic combination over encoded elliptic-curve group elements, while recovery of the encoded result is restricted to an authorized key holder [26-29]. The cloud therefore combines only the ciphertext-domain components defined in Equations (10)-(17); arbitrary ciphertext circuits are outside the scope of the proposed workflow.

Encode the processed financial plaintext information fragment a_i' and embed it into a random point $\dot{Q}_{a_i'}$ on elliptic curve B. Select a random number s_i and calculate the ciphertext (u_{1i}, u_{2i}) of the financial plaintext information fragment a_i' as follows:

$$\begin{cases} u_{1i} = s_i B_1 \\ u_{2i} = s_i B_2 + \dot{Q}_{a_i'} \end{cases} \tag{10}$$

This yields the ciphertext (u_{1i}, u_{2i}) of financial plaintext fragment a_i'. Decrypting the ciphertext (u_{1i}, u_{2i}) of this financial information fragment using the private key x yields:

$$\begin{cases} x u_{1i} = x(s_i B_1) = s_i B_2 = u_{2i} \\ u_{3i} u_{2i}^{-1} = a_i' \end{cases} \tag{11}$$

This yields the decrypted plaintext financial information a_i'. Perform operations on each plaintext financial information fragment a_i' to obtain ciphertext u_{1i}, u_{2i}, u_{3i}. Perform multiplication on all ciphertext $(u_{11}, u_{21}, u_{31}) \cdots (u_{1m}, u_{2m}, u_{3m})$ to obtain $(u_{11} \cdot u_{12} \cdots u_{1m}, u_{21} \cdot u_{22} \cdots u_{2m}, u_{31} \cdot u_{32} \cdots u_{3m})$. Using the private key x, perform the operation $x^m \cdot u_{11} \cdot u_{12} \cdots u_{1m}$ according to the upper equation in formula (11) to obtain:

$$x^m \cdot u_{11} \cdot u_{12} \cdots u_{1m} = x^m B_1{}^m s_1 \cdot s_2 \cdots s_m = B_2{}^m s_1 \cdot s_2 \cdots s_m = u_{21} \cdot u_{22} \cdots u_{2m} \tag{12}$$

Then, according to the lower formula in (11):

$$u_{31} \cdot u_{32} \cdots u_{3m} \cdot u_{21}{}^{-1} \cdot u_{22}{}^{-1} \cdots u_{2m}{}^{-1} = a_1' \cdot a_2' \cdots a_m' \tag{13}$$

This allows us to obtain the decrypted plaintext information $a_1' \cdot a_2' \cdots a_m'$ for all encrypted financial information.

2.2. Ciphertext Aggregation and Key Management for Fragmented Financial Data

2.2.1 Key Distribution and Trust Separation

A trusted key-management service initializes the elliptic-curve parameters and key group, distributes the required public/participant key material, and keeps aggregate decryption authority logically isolated from ordinary Map and Reduce workers. Edge terminals use authorized key material to protect shard outputs before transmission. General cloud workers receive only the material required for ciphertext processing, whereas final decryption is performed through the trusted key service after aggregation. Key identifiers, usage periods, protection requirements, and revocation status are bound to shard metadata so that expired or revoked keys cannot be silently reused. This separation is consistent with secure cloud key-management research and established lifecycle guidance for cryptographic keying material [33,34].

2.2.2 Reduce-Node Ciphertext Matching, Aggregation, and Verification

In secure financial-data aggregation, the Reduce stage performs ECC-based protected aggregation rather than unrestricted fully homomorphic computation [26-29]. A trusted key-management service within the MapReduce security domain holds the final aggregate-decryption authority, while ordinary Map/Reduce workers operate on protected shard values. When aggregating, the trusted manager determines the parameters and base point B_1 of elliptic curve B and broadcasts them, generating the key group $x_0, x_1, \cdots, x_m$, where $x_0 = x_1 + x_2 + \cdots + x_m$. The manager transmits the key x_0 to the MapReduce cloud computing center and sends the key $x_1 \sim x_m$ to the user $w_1 \sim w_m$. The MapReduce cloud computing center generates the private key x, calculates the public key B_2, and broadcasts it. User $w_1 \sim w_m$ encrypts the financial plaintext information using the public key B_2 and the key $x_1 \sim x_m$, and then sends the encrypted financial information ciphertext to the MapReduce cloud computing center, which completes the decryption and aggregation of the financial information ciphertext. The users w_i of the MapReduce cloud computing center refer to formula (10) and, by selecting a random number s_i, embed the multiplication results of each shard of financial plaintext information a_i' and the key x_i as points $\dot{Q}_{a_i'x_i}$ on the elliptic curve. After calculating the corresponding ciphertext financial information using the public key B_2, it is sent to the MapReduce cloud computing center. The MapReduce cloud computing center aggregates the ciphertext financial information received from each cloud user w_i, resulting in:

Building on protected aggregation and privacy-preserving distributed-computing principles [17,18,29], this study designs the following Reduce-node workflow. Step 1 (matching): every Map output is bound to a tuple consisting of data-set identifier, schema/feature identifier, time-window identifier, shard sequence, and key identifier; only ciphertexts with compatible tuples enter the same Reduce group. Step 2 (integrity screening): the Reduce worker verifies the transport authentication tag and rejects duplicated, stale, or malformed shard records. Step 3 (ciphertext aggregation): the worker combines accepted ciphertext components according to Equations (14)-(16) without exposing individual plaintext shards. Step 4 (authorized verification): the aggregate is passed to the trusted key service, where Equation (17) is evaluated and the recovered result is checked against shard count, metadata digest, and range constraints. The four workflow steps are design elements of the proposed framework rather than procedures attributed to the cited studies.

$$U = x\sum_{i=1}^{m} u_{1i} \tag{14}$$

Combining formula (10) yields:

$$U = x\sum_{i=1}^{m} u_{1i} = xB_1\sum_{i=1}^{m} s_i = B_2\sum_{i=1}^{m} s_i \tag{15}$$

Based on this, it can be concluded that:

$$\sum_{i=1}^{m} u_{2i} - U = B_2\sum_{i=1}^{m} s_i + \sum_{i=1}^{m} \dot{Q}_{a_i'x_i} - B_2\sum_{i=1}^{m} s_i = \sum_{i=1}^{m} \dot{Q}_{a_i'x_i} \tag{16}$$

After decrypting $\sum_{i=1}^{m}\dot{Q}_{a_i'x_i}$, $\sum_{i=1}^{m} a_i'x_i$ can be obtained. According to $x_0 = x_1 + x_2 + \cdots + x_m$, the final aggregation result of the financial information after encryption and decryption processing at the MapReduce cloud computing center is:

$$\sum_{i=1}^{m} a_i'x_i - x_0 = a_1' + a_2' + \cdots + a_m' \tag{17}$$

2.3. Distributed Storage of Financial Information Big Data Based on HDFS

After the MapReduce cloud computing center performs intelligent processing such as sharding, parallel fusion, encryption and decryption, and aggregation on the financial information, due to the large volume of financial data, it is necessary to use HDFS in the Hadoop cloud computing platform for distributed storage of the financial information processed by MapReduce, providing convenience for subsequent practical application analysis or queries of the financial information.

A typical HDFS consists of multiple Racks, with each Rack containing several storage servers (DataNodes) that enable high-speed financial information exchange through the internal network of the data center. The distributed storage structure of HDFS for financial information is shown in Figure 5.

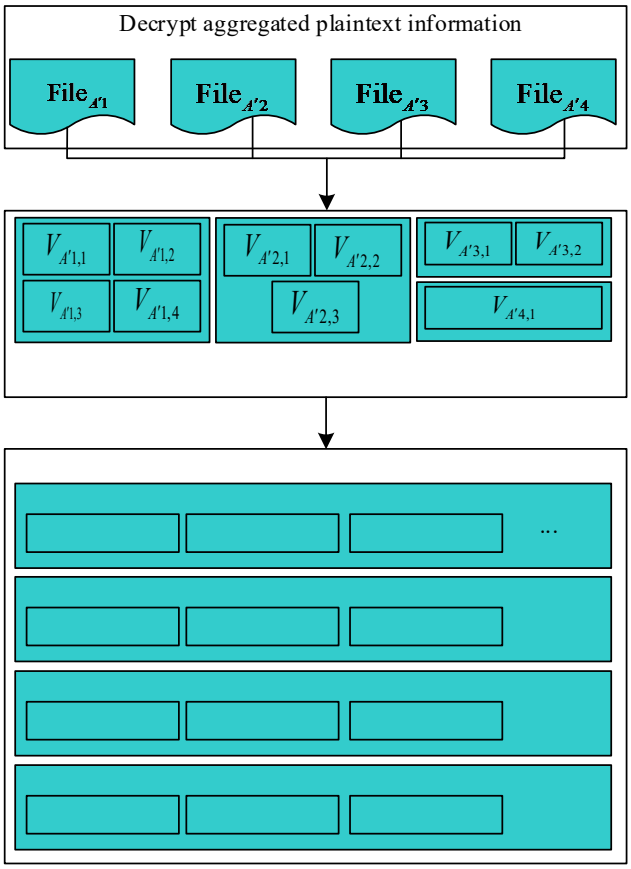


Figure 5. Structure diagram of HDFS for distributed storage of financial information

In the experimental HDFS configuration, each financial information file is divided into 64-MB blocks; 64 MB is treated here as an explicit experiment setting rather than as the default value of Hadoop 3.3.4. To enhance data reliability, HDFS stores multiple replicas of the same encrypted financial-information block and places them across DataNodes using the rack-aware replication strategy [23].

Definition 1 (HDFS Rack-Aware Storage Strategy): The first replica of the first financial information block is randomly stored in a certain data node, the second replica is stored in any data node on a different rack from the first financial information block, and the third replica is stored in the same rack as the second replica but in a different data node; if the replication factor of the financial information file is >3, the remaining financial information blocks are randomly stored in any data node other than the aforementioned three data nodes.

2.3.1 Encrypted-Block Replication, Failure Recovery, and Consistency

Encrypted financial blocks use replication factor r=3 unless a higher factor is configured. The first replica is placed on one DataNode, the second on a different rack, and the third on a different DataNode in the second replica's rack, consistent with the rack-aware rule in Definition 1 [23]. DataNodes periodically send heartbeats and block reports to the NameNode. When a heartbeat timeout marks a DataNode unavailable, the NameNode selects a healthy replica, schedules re-replication to a target that restores both the replication factor and rack diversity, and updates the block-location map only after the new replica is acknowledged. Each encrypted block is stored with a block identifier, generation/version metadata, and checksum. A recovered or replicated block is accepted only if the checksum and generation metadata match; stale or corrupt replicas are discarded and regenerated from a healthy copy. Because HDFS stores ciphertext rather than plaintext, recovery restores availability without requiring decryption inside the storage cluster.

Figure 5 shows that the client writes four financial information files $\mathrm{File}_{A'1} \sim \mathrm{File}_{A'4}$ into the HDFS cluster, with each financial information file being divided into multiple financial information blocks based on the size of the file. For example, $\mathrm{File}_{A'1}$ is divided into $V_{A'1,1} \sim V_{A'1,4}$ four financial information blocks, $\mathrm{File}_{A'2}$ is divided into $V_{A'2,1} \sim V_{A'2,3}$ three financial information blocks, $\mathrm{File}_{A'3}$ is divided into $V_{A'3,1}$、$V_{A'3,2}$ wo financial information blocks, while $\mathrm{File}_{A'4}$ is smaller than 64M, thus retaining one financial information block $V_{A'4,1}$. Each financial information block is placed in different DataNode nodes in the cluster following the HDFS rack-aware storage strategy.

The financial information segmentation and multi-replica storage method of HDFS forms the relationship between financial information blocks (replicas), files, DataNode nodes, and Racks. While a traditional simple graph can accurately represent binary relationships between information, it becomes cumbersome when representing the complex many-to-many mapping relationships among financial information files, financial information blocks, and DataNode nodes. Therefore, this paper proposes using hypergraph theory to establish a hypergraph model of the HDFS storage structure.

Definition 2 (financial information hypergraph): Let $Z=\{z_1,z_2,\cdots,z_n\}$ be a finite set of storage server DataNode nodes. If $V_i \neq \vartheta(i=1,2,\cdots,m)$ and $\bigcup_{i=1}^{m} V = Z$, then the binary relation $K=(Z,V)$ is called a hypergraph, where the elements $z_1,z_2,\cdots,z_n$ of Z are called the

vertices of the hypergraph, and $V=\{v_1,v_2,\cdots,v_n\}$ is the edge set of the hypergraph.

In the hypergraph model of the financial information HDFS storage structure, the mapping of the DataNode node set $Z=\{z_1,z_2,\cdots,z_n\}$ to the vertex set Z of this hypergraph model K, and the types of financial information blocks are mapped to the hyperedges V of this hypergraph K, thus the many-to-many relationship between financial information blocks and DataNode nodes can be represented by the hypergraph as:

(i) Financial information blocks and their replicas are stored on different DataNode nodes, which can be represented as the relationship $V\supset\{z_i,z_j,\cdots,z_r\}$ between the hyperedge of this type of financial information block and the nodes, where z_i represents the DataNode node storing the financial information, and $z_i\in Z$. The number of DataNode nodes contained in the hyperedge V is called the rank of the hyperedge, denoted as $\lambda(V)$, which is the number of different DataNode nodes occupied by the financial information block and its replicas.

(ii) The different types of financial information blocks stored in a certain DataNode node z_i correspond to the number of hyperedges connected to vertex z_i in the hypergraph K, which is called the degree of vertex z_i, denoted as $K(z_i)$.

Definition 3 (Financial Information Hypergraph Adjacency Matrix): The matrix $\boldsymbol{H}$ represents the adjacency matrix of the financial information hypergraph $K=(Z,V)$. Through the mathematical expression implemented by $\boldsymbol{H}(\omega_{ij})$, it can be concluded that the m column in the financial information hypergraph corresponds to the m hyperedges of hypergraph $K=(Z,V)$, which correspond to the n vertices $Z=\{z_1,z_2,\cdots,z_n\}$ contained in the financial information hypergraph. When $z_i\in V_j$, $\omega_{ij}=1$; conversely, when $z_i\notin V_j$, $\omega_{ij}=0$. Thus, according to Definition 2, the node degree $K(z_i)$ of the DataNode node z_i in the financial information hypergraph is:

$$K(z_i)=\sum_{j=1}^{m}\omega_{ij} \tag{18}$$

By implementing intelligent processing such as integration, encryption, and aggregation on massive financial information big data through this HDFS financial information distributed storage hypergraph model, it provides assurance for the security and integrity of subsequent financial information applications.

2.4. Cloud-Edge Collaborative Task Allocation and Secure Transmission

The cloud-edge deployment assigns privacy-sensitive preprocessing to edge financial terminals and compute-intensive aggregation/storage to the cloud. Edge terminals perform source-system access, schema validation, local shard construction, feature normalization required by the RF module, and ECC-based protection. Only protected shard outputs and non-sensitive routing metadata are transmitted to cloud Map workers. Cloud nodes execute parallel Map/Shuffle/Reduce processing and HDFS storage, while the trusted key-management domain remains logically separated from both edge application processes and cloud aggregation services. This division is consistent with data-protection-aware cloud-edge offloading, finance-oriented edge security, established edge-computing architectures, and heterogeneous-cloud scheduling principles [20,21,30-32].

For edge-cloud transmission, each shard is encapsulated with shard ID, schema ID, key ID, timestamp/nonce, and an integrity tag, and the channel is mutually authenticated. Replay protection is enforced through the nonce and freshness window, while rejected or missing shards are retransmitted before the Reduce barrier is released. Cross-region deployment can keep raw financial records and private keys within the originating jurisdiction and transfer only protected shards when permitted by the applicable policy. This mechanism supports data-localization and audit requirements but is not presented as a substitute for jurisdiction-specific legal review.

2.5. Nomenclature

The cluster configuration and key parameters for simulation experiments in this paper are shown in Table 2.

Table 2. Nomenclature of the main variables, parameters, and system terms

Symbol / term	Definition
A; ai	Raw multi-source financial collection; i-th financial-data subset/shard
m; i; j	Number of subsets; shard index; decision-tree/worker index
f(·); lai	Mapping/inference function; size of shard ai
L; K	Total raw-data size; plaintext-bit expansion/sharding parameter
pos; filebuffer	Read position; temporary shard buffer
dj; ωj; D	j-th decision tree; its random parameter vector; random-forest ensemble
ep[p|ai]; ep[p|A]	Shard-level and collection-level classification evaluation probability
s(ai,p); μ; δ	Fusion boundary function; generalization-error value; expected classification evaluation
CK(·); DK(·)	Encryption and decryption transformations in the protected workflow
B; B1; B2; x	Elliptic curve; base point; public point/key; private scalar/key
Q, G, Δ	Encoded point; curve-intersection point; line/tangent slope
si; (u1i,u2i)	Per-shard randomizer; ECC ciphertext components
xi; x0; wi	User/key share terms and aggregation participant index used in Eqs. (14)-(17)
V; E; H	DataNode vertex set; hyperedge set; financial-information hypergraph
r	HDFS encrypted-block replication factor
PLEI	Plaintext Leakage Exposure Index: plaintext crossings of untrusted boundaries / sensitive crossings
Te2e; Qp	End-to-end processing latency; parallel throughput

3. Security, Compliance, and Industrial Deployment Analysis

3.1. Threat Model and Privacy Boundary

The threat model assumes that network links can be eavesdropped, cloud workers can be honest-but-curious, and individual DataNodes can fail or be read by an unauthorized infrastructure operator. Edge data owners and the trusted key-management service are assumed not to be simultaneously compromised. Under this model, the security objective is to prevent plaintext financial shards and decryption keys from appearing at untrusted cloud processing or storage points, while maintaining availability under node/rack failures. The model does not claim resistance to a fully compromised authorized endpoint or to side-channel attacks on the cryptographic implementation; these remain deployment-level risks.

3.2. Compliance-Oriented Deployment and Long-Term Impact

For data-localization requirements, the architecture can confine raw records and local identifiers to edge infrastructure operated within the required jurisdiction and place aggregate private-key custody in a jurisdiction-controlled key-management service, while ordinary cloud workers receive only protected shards. For privacy audits, shard/key identifiers, aggregation-group identifiers, access events, replication events, and recovery events can be logged without storing plaintext transaction content. For cross-regional processing, policy gates can block shard export when the destination region or key policy is not authorized. These controls support regulatory implementation principles such as data minimization, separation of duties, auditability, and localization, but final compliance depends on the financial institution's applicable laws, retention rules, and governance procedures [12,15-16].

The long-term deployment impact can be interpreted through a normalized cost model $C_{total} = C_{compute} + C_{network} + C_{storage} + C_{audit}$. Map parallelism reduces $C_{compute}$ per completed shard until coordination overhead dominates; edge-local filtering and protected shard transfer reduce unnecessary cross-boundary traffic; HDFS replication increases $C_{storage}$ but improves recovery and service availability; and explicit key/audit separation reduces the operational cost of investigating privacy events. The experiment identifies 42 Map nodes as the best observed processing point for the tested workload, providing a practical capacity-planning reference rather than assuming that adding workers always reduces total cost.

4. Experimental Results and Reproducibility Analysis

4.1. Dataset, Baselines, and Evaluation Metrics

The evaluation uses a 5.5-TB operational financial-information dataset from a large e-commerce company. Because the source data are commercially confidential, the raw dataset cannot be publicly redistributed; Table 3 reports only representative non-identifying transaction fields available in the manuscript. The processing pipeline is deterministic at the system level: input records are validated against the financial schema, partitioned by the sharding function, processed by RF Map workers, protected by the ECC-based encoding workflow, aggregated by Reduce workers, and stored as encrypted HDFS blocks. The 5-GB subset used for the scalability experiment is divided into a1=0.5 GB, a2=1.5 GB, and a3=3 GB.

Table 3. Example of partial financial information in the experimental financial information big data set

Date	Income Type	Income Amount (CNY)	Expense Type	Expense Amount (CNY)
2024/5/1	Product Sales	5000	Procurement Cost	3000
2024/5/2	Product Sales	4500	Logistics Costs	800
2024/5/3	Product Sales	6000	Employee Salaries	2000
2024/5/4	Advertising Income	1000	Office Equipment Procurement	1500
2024/5/5	Product Sales	5500	Utility Bills	300
2024/5/6	Product Sales	4800	Marketing Expenses	1200
2024/5/7	Product Sales	5200	Rent	2500

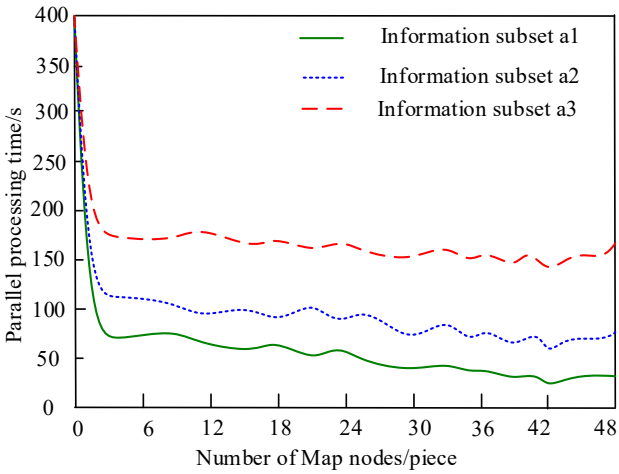
Table 4. Baselines and unified evaluation metrics

Item	Definition / implementation
Proposed MapReduce-ECC	Full sharding, RF fusion, ECC-based protection, Reduce aggregation, HDFS storage
Single-node ECC ablation	Same protected workflow with one Map worker; the one-worker point in Figure 6 is the reference condition
Blockchain + FL [6]	System-level privacy architecture implemented from the blockchain/federated-learning design principles in [6]; not used for raw key-length equivalence claims

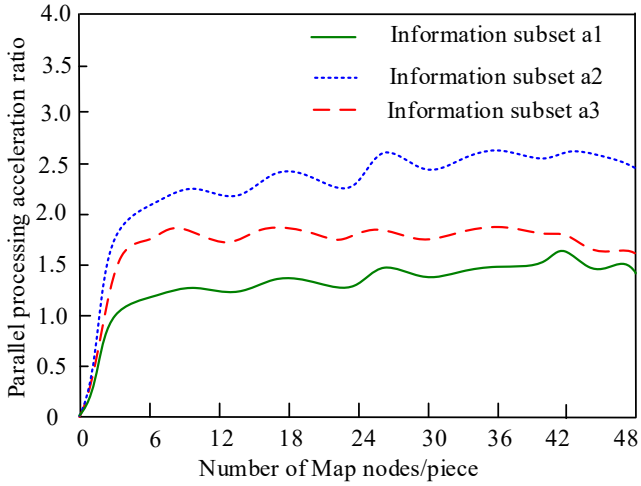
Blockchain secure storage [7]	System-level supply-chain-finance storage/access-control configuration inspired by [7]; not treated as a standalone cryptographic primitive
RDMA/NVRAM [8]	Storage/communication reference configuration inspired by the RDMA/NVRAM design in [8]; not an encryption algorithm
End-to-end latency Te2e	Shard + fusion + encryption + shuffle + aggregation + storage time
Encryption overhead Ωenc	Relative additional processing time introduced by protected processing at a fixed data volume/security setting
Parallel throughput Qp	Processed financial-data volume divided by elapsed processing time
PLEI	Fraction of sensitive trust-boundary crossings that expose plaintext; lower is better
Storage reliability	Task-completion ratio, post-storage availability ratio, and storage time

4.2. Parallel Scalability, Latency, and Throughput

Parallel scalability is evaluated by varying the number of Map workers for the three data subsets. The one-Map-worker condition is treated as the single-node ECC ablation, while larger worker counts activate the proposed parallel processing path. Processing time provides the direct latency measure, and the acceleration ratio in Figure 6 is the inverse indicator of normalized latency; throughput increases as the same data volume is completed in less time. This design makes the single-node versus parallel comparison explicit without introducing an additional unverified dataset.



(a) Parallel fusion encryption processing time statistics



(b) Statistics on acceleration ratio of parallel fusion encryption processing

Figure 6. Parallel processing latency and acceleration ratio under different numbers of Map workers

As shown in Figure 6, increasing the number of Map workers substantially reduces processing latency at low-to-moderate worker counts and increases the acceleration ratio relative to the single-worker ECC reference. Beyond a threshold, coordination, shuffle, and scheduling overhead offset the benefit of additional workers. For the tested workload, 42 Map workers provide the best observed trade-off, after which processing time and acceleration remain

approximately stable. This result supplies a quantitative scalability boundary for deployment planning rather than assuming linear speedup.

4.3. Cryptographic Overhead, Security-Strength Interpretation, and Plaintext-Exposure Analysis

The blockchain/federated-learning, blockchain secure-storage, and RDMA/NVRAM configurations in Table 4 are heterogeneous system-level references [6-8]. They are used to discuss distributed privacy, storage, and communication behavior where technically comparable, but they are not treated as cryptographic primitives with directly comparable key lengths. Encryption overhead of the implemented ECC-based path is therefore evaluated through the one-worker versus multi-worker conditions in Figure 6, which isolate the effect of MapReduce parallelism on the same protected-processing workflow.

Cryptographic security is interpreted by equivalent security strength rather than raw key-bit counts. NIST guidance associates approximately 112-bit security with 2048-bit RSA and 224-255-bit ECC, and approximately 128-bit security with 3072-bit RSA and 256-383-bit ECC [34]. Accordingly, this study does not infer cryptographic superiority from a direct comparison of heterogeneous raw key lengths. The present experiment reports processing latency/acceleration for the implemented ECC-based path in Figure 6 and evaluates the Plaintext Leakage Exposure Index (PLEI) at protocol level. PLEI is the number of sensitive trust-boundary crossings that carry plaintext divided by the total number of sensitive crossings. Under the stated cloud-edge rule, raw records remain inside the trusted edge boundary and edge-cloud/HDFS transfers contain protected shard data, so the designed cross-boundary plaintext exposure is zero. PLEI is therefore an architectural exposure metric, not an empirical attack-success probability. A matched-security benchmark across RSA, Paillier, and EC-ElGamal implementations is left for future controlled evaluation.

4.4. HDFS Deployment, Fault Tolerance, and Storage Performance

To evaluate the storage layer, the encrypted-data HDFS cluster contains seven racks and eight DataNodes per rack (56 DataNodes in total). The experiment measures load balance, storage-task completion, post-storage availability, and storage time. The rack-aware recovery logic defined in Section 2.3.1 is part of the deployment configuration; the present experimental material validates storage balance and availability but does not contain separate run-level rack-failure recovery latency traces, which is identified as a limitation rather than retrospectively fabricated.

Table 5. DataNode node configuration parameters

Parameter	Set Value
CPU	AMD Ryzen 5 5600X 3.7GHz / 6 cores
OS Version	CentOS 8.4
Hard Disk	NVMe SSD (PCIe 4.0 x4, 7000MB/s Read Speed)
Java Version	Java 11 for Linux
Hadoop Version	3.3.4
Minimum Operating Power Consumption of DataNode	120W
Maximum Operating Power Consumption of DataNode	280W

The physical scene diagram of the HDFS cluster used in the experiment is shown in Figure 7.



Figure 7. Physical scene of HDFS cluster used in the experiment

The distributed storage load balancing performance of the HDFS cluster used in this paper was tested. The experimental financial information big data set processed as described above was distributed across the HDFS cluster using the method proposed in this paper. Taking the three financial information subsets a1, a2, and a3 as examples, the standard deviation of the information storage volume across the DataNode nodes in the HDFS cluster was calculated. Based on the statistical results, the distributed storage load balancing of the method proposed in this paper was verified, with the results shown in Figure 8.

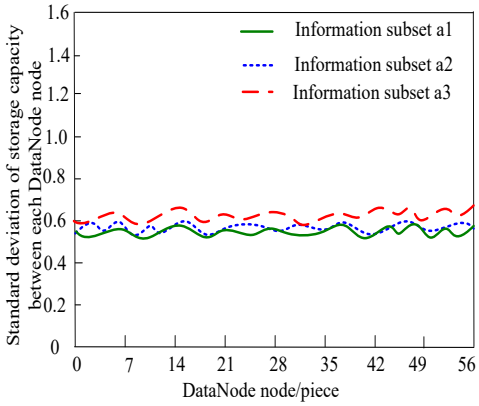
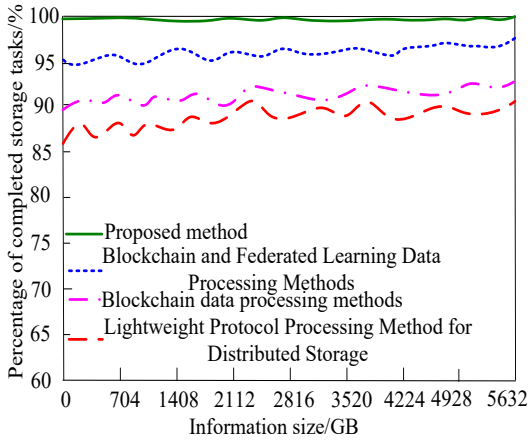


Figure 8. Results of distributed storage load balancing test for the method proposed in this article

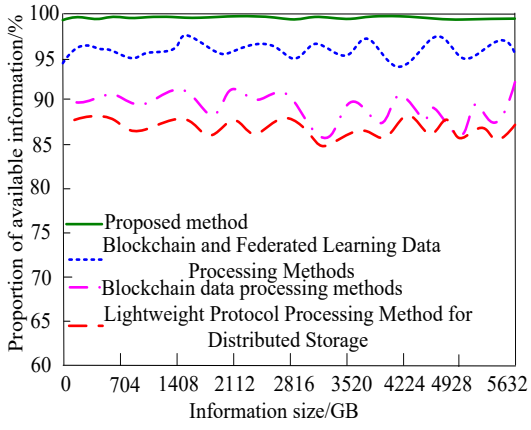
As shown in Figure 8, during the distributed storage experiment of financial information using the HDFS cluster proposed in this article, the standard deviation of the information storage volume among the DataNode nodes shows only small differences, with all values within 0.7 across the evaluated information volumes and DataNode counts. This indicates that the HDFS cluster used in the experiment maintains a relatively balanced distribution of financial information across DataNodes, supporting the effectiveness of the distributed-storage configuration.

Ensuring the storage effectiveness and post-storage availability of financial information is a critical task in the intelligent processing of financial information. To this end, this study further evaluates the storage performance of various methods by comparing their task completion rates, post-storage availability of financial information, and storage duration when handling large volumes of financial information in a centralized manner. The results of this comparison are shown in Figure 9.

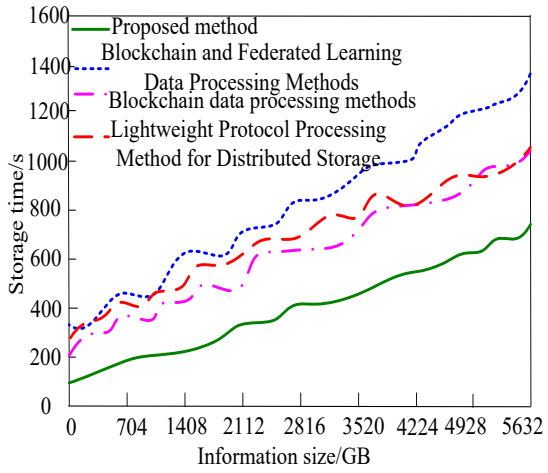
For the system-level comparison in Figure 9, the non-proposed configurations are local testbed adaptations inspired by the architectural categories discussed in [6-8], rather than reproductions of those publications. The cited studies use different datasets, objectives, hardware, and software stacks; therefore, the numerical values in Figure 9 are measurements from the present HDFS testbed and must not be interpreted as evidence that the proposed method empirically outperforms the original published systems.



(a) Comparison of completion status of financial information storage tasks



(b) Comparison of information availability after financial information storage



(c) Comparison of storage time for financial information

Figure 9. Comparison of financial information storage performance of various methods

As shown in Figure 9, within the present adapted testbed configurations, the proposed method maintains storage-task completion and post-storage availability close to 100% across the evaluated data volumes. Its storage time also grows more slowly than those of the comparison configurations in this testbed. The comparison configurations show lower task completion and availability ratios and larger storage-time growth under the tested settings. These results support the storage efficiency of the proposed configuration in the present experimental environment; because the comparison configurations are adaptations rather than exact reproductions of [6-8], the result is not generalized to the original published systems.

5. Conclusion

This paper develops a privacy-preserving cloud-edge financial information processing framework that couples MapReduce sharding and RF fusion with ECC-based protected aggregation and encrypted HDFS storage. The framework makes key separation, Reduce-node ciphertext matching, authorized decryption verification, cloud-edge task allocation, and HDFS recovery explicit. In the 5.5-TB operational-data evaluation, the 5-GB scalability subset shows that 42 Map workers provide the best observed parallel-processing trade-off. The HDFS experiments maintain storage-task completion and post-storage availability close to 100% while preserving balanced DataNode utilization. Cryptographic comparisons are interpreted by equivalent security strength rather than raw key-bit counts, and the present experiments do not claim a matched-security superiority over unrelated blockchain or RDMA system architectures. For retail banks, the recommended deployment is to keep raw customer/transaction records at jurisdiction-controlled edge infrastructure and keep aggregate decryption keys in a jurisdiction-controlled, preferably hardware-backed key-management service, while general cloud workers handle only protected aggregation and replicated storage. For internet-fintech platforms, Map-worker elasticity can be used for burst workloads while key management and audit logs remain separated from cloud compute. For regulators and internal auditors, shard IDs, key IDs, aggregation events, replication events, and recovery records provide an auditable processing trail without requiring plaintext transaction content. The framework supports, but does not by itself guarantee, compliance with data-localization, privacy-audit, and retention requirements. Remaining limitations include the absence of public redistribution rights for the operational dataset, lack of run-level rack-failure recovery timing, and the need for controlled multi-user concurrency and matched-security cryptographic benchmarking. Future work will evaluate threshold-key management, hardware-backed edge key custody, concurrent ciphertext aggregation, cryptographic baselines at matched NIST security strengths, and recovery behavior under injected node/rack failures.

Acknowledgements

The authors would like to thank the editors and anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper.

References

[1] Yu X. Study of corporate management financial early warning combining BP algorithm and KLR. Int J Crit Infrastruct. 2025; 21(1):87-103.
[2] Tong D, Tian G. Intelligent financial decision support system based on big data. J Intell Syst. 2023; 32(1):20220320.
[3] Bi S. Reliability analysis of the financial management auxiliary system of remote branches based on cloud computing. Int J Appl Syst Stud. 2023; 10(2):136-153.
[4] Duan Y. Design of accounting information big data analysis platform based on cloud computing. Procedia Comput Sci. 2024; 247:1128-1136.
[5] Noshari MR, Azgomi H, Asghari A. Efficient clustering in data mining applications based on harmony search and K-medoids. Soft Comput. 2024; 28(23-24):13245-13268.
[6] Kang H, Wu B. Enhancing data security in massive data sets using blockchain and federated learning: a loosely coupled approach. Int J Internet Protoc Technol. 2024; 17(1):31-41.
[7] Li D, Han D, Crespi N, Minerva R, Li KC. A blockchain-based secure storage and access control scheme for supply chain finance. J Supercomput. 2023; 79(1):109-138.
[8] Liu Q, Varman PJ. Telepathy: a lightweight silent data access protocol for NVRAM+RDMA enabled distributed storage. IEEE Trans Comput. 2023; 72(3):839-852.
[9] Gupta M, Dwivedi RK. Fortified MapReduce Layer: Elevating Security and Privacy in Big Data. EAI Endorsed Scal Inf Syst. 2023; 10(6).
[10] Tian X. Generative AI-based privacy protection and security management mechanism for distributed financial systems. EAI Endorsed Scal Inf Syst. 2026; 12(10).

[11] Naresh VS, Ayyappa D. PPDNN-CRP: privacy-preserving deep neural network processing for credit risk prediction in cloud: a homomorphic encryption-based approach. J Cloud Comput. 2024; 13:149.
[12] Ren YS, Ma C, Wang Y. A new financial regulatory framework for digital finance: inspired by CBDC. Glob Finance J. 2024; 62:101025.
[13] Shafin KM, Reno S. Enhancing foreign exchange reserve security for central banks using Blockchain, FHE, and AWS. J King Saud Univ Comput Inf Sci. 2024; 36(10):102251.
[14] Guo S. Federated learning financial data management techniques based on data distillation and dynamic tiered scheduling. Discov Appl Sci. 2025; 7:1406.
[15] Alaklabi S. PrivChain-AI leveraging blockchain and federated learning for private financial reporting and access control. Sci Rep. 2026; 16:2786.
[16] Naresh VS, Ayyappa D. Privacy-preserving federated credit risk models: evaluating differential privacy and homomorphic encryption techniques. Sci Rep. 2026; 16:4379.
[17] Jiang S, Yang H, Xie Q, Ma C, Wang S, Liu Z, Xiang T, Xing G. Towards compute-efficient Byzantine-robust federated learning with fully homomorphic encryption. Nat Mach Intell. 2025; 7:1657-1668.
[18] Anitha R, Murugan M. Privacy-preserving collaboration in blockchain-enabled IoT: the synergy of modified homomorphic encryption and federated learning. Int J Commun Syst. 2024; 37(18):e5955.
[19] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. Commun ACM. 2010; 53(4):50-58.
[20] Mann ZA, Metzger A, Prade J, Seidl R, Pohl K. Cost-optimized, data-protection-aware offloading between an edge data center and the cloud. IEEE Trans Serv Comput. 2023; 16(1):206-220.
[21] Luo Y. Financial data security management method and edge computing platform based on intelligent edge computing and big data. IETE J Res. 2023; 69(8):5187-5195.
[22] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM. 2008; 51(1):107-113.
[23] Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies; 2010. p. 1-10.
[24] Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies; 2010; Location of conference. Place of publication: Publisher's name; p. 1-10.
[25] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. EUROCRYPT 1999. Lecture Notes in Computer Science (volume 1592); 1999; Location of conference. Place of publication: Publisher's name; p. 223-238.
[26] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory. 1985; 31(4):469-472.
[27] Koblitz N. Elliptic curve cryptosystems. Math Comput. 1987; 48(177):203-209.
[28] Miller VS. Use of elliptic curves in cryptography. Advances in Cryptology - CRYPTO '85. Lecture Notes in Computer Science (volume 218); 1986; Location of conference. Place of publication: Publisher's name; p. 223-238.417-426. .
[29] Li L, Abd El-Latif AA, Niu X. Elliptic curve ElGamal based homomorphic image encryption scheme for sharing secret images. Signal Process. 2012; 92(4):1069-1078.
[30] Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE Internet Things J. 2016; 3(5):637-646.
[31] Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor. 2017; 19(4):2322-2358.
[32] Ayyadi A, Jahani A. Addressing cost and resource variability for big data task scheduling in heterogeneous cloud environments. J Supercomput. 2025; 81:762.
[33] Ahmad S, Mehfuz S, Urooj S, Alsubaie N. Machine learning-based intelligent security framework for secure cloud key management. Cluster Comput. 2024; 27(5):5953-5979.
[34] Barker E. Recommendation for Key Management: Part 1 - General. NIST Special Publication 800-57 Part 1 Rev. 5. Gaithersburg (MD): National Institute of Standards and Technology; 2020.