MF-BFDSA: A Multi-Feature Fusion and Dynamic Weight Algorithm for Real-Time Blockchain Financial Transaction Risk Detection

Zhiyuan Xiao
Business School , Central South University, Hunan Changsha 410083, China

Keywords: blockchain finance, transaction security, multi-feature fusion, dynamic weight adjustment, MF-BFDSA algorithm, risk detection, hyperledger fabric

Received: September 18, 2025

To address the challenges of insufficient accuracy in identifying complex risks in blockchain financial transactions and high detection latency in high-concurrency scenarios, this paper constructs a multi-feature fusion dynamic security detection algorithm (MF-BFDSA) and designs a security enhancement mechanism. This algorithm innovatively integrates three types of features: transaction data (amount, frequency, etc.), user behavior, and blockchain network (node response latency, etc.). It uses Q-learning to dynamically adjust weights and combines an improved LSTM (using a forget gate to introduce feature correlation coefficients) with an optimized XGBoost algorithm to construct a risk identification model. This mechanism utilizes a five-layer architecture encompassing data collection and algorithm detection, forming a dual barrier of "algorithm pre-detection + blockchain strong verification." Experiments were conducted on the Hyperledger Fabric 2.4 platform, using Kaggle credit card fraud (284,000 records), IEEE-CIS financial fraud (550,000 records), and 1.05 million simulated transaction data. Comparing traditional algorithms with state-of-the-art methods, the results show that MF-BFDSA achieves an accuracy of 99.82% (surpassing state-of-the-art by 1.2-3.5 percentage points), an F1 score of 98.94% (surpassing state-of-the-art by 2.8-4.2 percentage points), and an anomaly detection latency of only 0.08 seconds. It also maintains a throughput of 1200 TPS and a latency of 0.15 seconds on mid-range hardware. A blockchain system integrating this algorithm achieves a throughput of 1800 TPS (50% higher than the original system) at 2000 TPS concurrency, a CPU utilization of 72% (reduced by 13 percentage points), and a double-spend success rate of less than 0.001%, validating the effectiveness of the algorithm and mechanism.

Povzetek: Članek predstavi MF-BFDSA, večznačilčni dinamični algoritem za zaznavanje tveganj v verigah blokov za visokonatančno in nizkolačno odkrivanje prevar pri visoki sočasnosti.

1 Introduction

As financial transactions rapidly develop towards digitalization and decentralization, security threats such as transaction data tampering, user identity forgery, and sensitive information leakage are becoming increasingly prominent, posing severe challenges to the stable operation of the financial system. Traditional financial security mechanisms rely on centralized authentication architectures, which have obvious limitations in decentralized transaction scenarios. Centralized nodes are vulnerable to attacks and respond slowly to real-time transaction risk identification, making it difficult to cope with high-frequency, complex, and new types of fraud. Blockchain technology has shown its potential in the field of financial transaction security due to its decentralized, tamper-proof, and traceable characteristics. It was initially applied to scenarios such as cross-border payments and supply chain finance. However, existing blockchain financial transaction systems still have significant security flaws, such as smart contract code vulnerabilities that are easily exploited (such as the reentrancy attack in The DAO incident), and the imbalance between efficiency and security in the consensus mechanism (the PoW mechanism has high energy consumption and low throughput, while the PoS mechanism faces the risk of "rich man attacks"). It is difficult to meet the dual requirements of efficiency and security for financial transactions [1]. Current blockchain financial security research mainly focuses on consensus mechanism optimization (such as improving the PBFT algorithm to reduce communication complexity and proposing a hybrid consensus model to improve fault tolerance) and privacy protection technology (such as zero-knowledge proof and homomorphic encryption to hide the identity and amount of transaction participants) [2]. However, these studies mostly focus on a single technical dimension and lack systematic integration; the application of intelligent algorithms in financial risk control focuses on machine learning anti-fraud and deep learning anomaly detection (such as building a transaction fraud classification model based on the SVM algorithm and

using the LSTM network to capture the time characteristics of user behavior) [3]. However, the degree of integration between blockchain and intelligent algorithms in existing research is limited, and a comprehensive security mechanism covering the entire process of transaction initiation, verification, and chain-up has not yet been established. In addition, intelligent algorithms often rely on single feature input. Even multi-feature research lacks clear innovation in weight adjustment logic (such as fixed weights based on experience). There is insufficient experimental verification on various real data sets, making it difficult to implement in actual financial scenarios. To this end, the research objectives set in this paper include: first, designing the MF-BFDSA algorithm, which is specifically used to address the problems of low accuracy and high latency in complex risk identification in blockchain financial transactions; second, building a security enhancement mechanism based on the MF-BFDSA algorithm to ensure security throughout the entire transaction process, from initiation to on-chain storage; and third, verifying the performance of the algorithm and mechanism on diverse datasets (including real and simulated datasets) and different hardware (including high-end, mid-range, and low-end hardware) to confirm its feasibility in actual deployment. The research questions corresponding to the research objectives are as follows: First, can multi-feature fusion (including transaction features, user features, and blockchain features) and dynamic weight adjustment based on reinforcement learning improve the accuracy of risk identification compared to single features or fixed weight algorithms? Second, what impact will the interaction of "algorithm pre-detection + blockchain verification" have on consensus efficiency, and what is the trade-off between security (measured by false rejection rate) and efficiency (measured by consensus latency)? Third, can the integrated system be deployed on mid- and low-end hardware, and what are the characteristics of its resource consumption (including CPU consumption and memory consumption) under typical banking business loads? Fourth, in addition to being able to deal with DDoS attacks and double-spending attacks, can this mechanism also address broader security challenges such as cross-chain privacy, regulatory compliance, and hybrid blockchain adaptability? The research technology route of this paper follows the logical process of “demand analysis → algorithm design → mechanism construction → experimental simulation → result analysis → conclusion and prospect” to ensure the systematic and rigorous research process. First, a blockchain financial transaction dynamic security detection algorithm (MF-BFDSA) based on multi-feature fusion is designed to achieve accurate real-time identification of transaction risks. Secondly, a blockchain financial transaction security enhancement mechanism is constructed based on this algorithm to form a comprehensive protection system of “data collection - dynamic detection - blockchain verification - security response”. Finally, the security and effectiveness of the mechanism are verified through experimental simulation [4].

2 Related technical foundations

2.1 Core blockchain technology

Blockchain technology achieves decentralized data storage and interaction through a layered architecture. Each of its five layers has clear functional boundaries and works in tandem. The data layer is the foundation, storing transaction records in blocks [5]. These records are linked together through hash values to form a chain structure, ensuring data immutability. The network layer uses a peer-to-peer (P2P) protocol for distributed communication between nodes, ensuring synchronized data transmission across multiple nodes. The consensus layer uses specific rules to achieve data consistency between nodes, a core guarantee of blockchain security. The contract layer supports the deployment of smart contracts, codifying transaction logic for automated execution. The application layer provides application access through APIs for financial transactions and other scenarios, such as blockchain payment platforms and supply chain finance systems.

The consensus mechanism determines the security and efficiency of a blockchain network, and the applicability of mainstream mechanisms in financial scenarios varies. The PoW (Proof of Work) mechanism verifies transactions through competitive node computing power, offering high security. However, it suffers from high energy consumption and long transaction confirmation delays (approximately 10 minutes per block), making it difficult to meet the demands of high-frequency financial transactions. The PoS (Proof of Stake) mechanism uses node token holdings as a basis for competition, reducing energy consumption and latency. Still, it can easily lead to "rich node monopolies" and pose security risks [6]. The PBFT (Practical Byzantine Fault Tolerance) mechanism reaches consensus through multiple rounds of node voting, offering low latency (in milliseconds) and high fault tolerance (tolerating 1/3 malicious nodes), making it more suitable for the real-time and security requirements of financial transactions. However, communication complexity increases as the number of nodes increases, limiting scalability.

Privacy protection technologies are key enablers of blockchain financial applications. Zero-knowledge proofs and homomorphic encryption are commonly used technologies, but have limitations. Zero-knowledge proofs can verify the legitimacy of transactions without revealing transaction details (such as the amount or the parties' identities). They are suitable for privacy-sensitive scenarios such as cross-border payments. However, the proof generation and verification process is computationally expensive, impacting transaction efficiency [7]. Homomorphic encryption allows operations on data in an encrypted state, ensuring privacy during data processing. It can be used for financial data sharing and risk management analysis, but currently only supports a limited number of operations (such as homomorphic addition) and suffers from slow encryption

and decryption speeds, making it difficult to adapt to large-scale transaction scenarios.

2.2 Key security requirements for financial transactions

The digital nature of financial transactions exposes them to multi-dimensional security threats, requiring them to meet four core requirements. Transaction integrity ensures that transaction data is not tampered with during transmission and storage. Malicious data modification (such as altering transaction amounts or recipient accounts) will lead to financial loss. Therefore, data immutability must be achieved through technologies such as hash verification and chain storage. Identity authenticity aims to verify the legitimacy of transaction participants and prevent identity forgery (such as using stolen accounts to initiate transactions). This requires user identity verification through digital signatures and multi-factor authentication. Risk in real time requires real-time identification and interception of transaction risks. Financial fraud (such as credit card theft and money laundering) is sudden and hidden, and delayed responses can exacerbate losses [8]. Therefore, real-time detection technology is required to identify risks within seconds. Data privacy protects sensitive information of transaction participants (such as names, bank card numbers, and transaction details) to prevent privacy risks and financial security issues caused by information leaks. Data desensitization requires encryption and anonymization.

2.3 Basic applications of intelligent algorithms in security

Machine learning algorithms provide basic classification and prediction capabilities for financial security detection, with decision trees and support vector machines being typical examples. Decision trees achieve classification by constructing a tree-like structure, dividing decision nodes based on characteristic attributes (such as transaction amount and frequency). This allows for intuitive output of risk assessment logic and is suitable for simple transaction risk assessment. However, they have limited feature extraction capabilities for high-dimensional data and are prone to overfitting. Support vector machines (SVMs) achieve data classification by finding an optimal hyperplane. They can effectively handle high-dimensional features (such as user behavior and transaction time series) and excel in detecting small-value payment fraud [9]. However, their training efficiency is low for large datasets, and the classification results are difficult to interpret.

Deep learning algorithms enhance feature extraction and pattern recognition capabilities with their deep network structures. CNNs and LSTMs are widely used in financial security. CNNs (convolutional neural networks) extract local data features through convolutional and pooling layers. They can mine hidden risk features (such as abnormal operation sequences) from unstructured data such as transaction logs and user operation records. They are suitable for risk identification in complex scenarios, but require large amounts of labeled data for training and are inadequate for capturing temporal features. LSTMs (long short-term memory networks) process time series data through a gating mechanism (input gate, forget gate, and output gate). They can effectively capture the temporal correlations between transactions (such as multiple inter-regional transactions within a short period) and effectively detect fraudulent transactions and identify continuous fraudulent activity. However, their model structure is complex, training cycles are long, and their response speed to short-term, sudden risks needs to be improved [10]. The application logic of intelligent algorithms in the financial security field can be divided into three steps: First, collect features such as transaction data (such as amount, time, and device information) and user behavior data (such as login IP address and operating habits); second, use algorithmic models (such as LSTM and SVM) to learn these features and build a risk identification model; finally, input real-time transaction data into the model, output a risk level (such as low risk or high risk), and trigger corresponding responses (such as releasing normal transactions and blocking high-risk transactions), forming a closed-loop application process of "data collection - model training - real-time detection."

2.4 Related works summary and comparison

Table 1 summarizes recent representative works on blockchain-based financial fraud detection, comparing their methods, datasets, performance, and limitations with MF-BFDSA. Existing works have three critical shortcomings: 1) **Feature Limitation**: Most rely on single-feature groups (e.g., only transaction data), failing to capture complex correlations between user behavior and blockchain network states; 2) **Efficiency-Security Trade-off**: Lack of clear interaction logic between algorithms and consensus mechanisms, leading to either low security (e.g., no additional verification for abnormal transactions) or high latency (e.g., all transactions require multi-node verification); 3) **Deployability Gap**: Performance validation is limited to high-performance clusters, ignoring medium/low-end hardware common in small financial institutions. MF-BFDSA addresses these by: integrating three feature groups, designing scenario-aware dynamic weights, optimizing algorithm-blockchain interaction, and verifying on diverse hardware.

Table 1: Recent representative works on blockchain-based financial fraud detection

Reference	Method	Dataset Type/Size	Accuracy (%)	F1 (%)	Key Limitations	MF-BFDSA Advantages
[2] Wu et al. (2024)	Blockchain + SVM (single transaction feature)	Real-world (300k records)	97.2	95.8	Single-feature input, low complex risk accuracy	Multi-feature fusion (3 types), 2.62% higher accuracy
[8] Bello et al. (2024)	Blockchain + Standard LSTM	Real-world (450k records)	98.1	96.3	Fixed weights, high latency (0.2s)	Dynamic weighting (RL-based), 0.12% higher accuracy, 0.12s lower latency
[17] Chaudhry et al. (2023)	Blockchain + GNN	Real-world (500k records)	98.6	97.1	High computational complexity, poor medium-hardware adaptability	Optimized complexity (O(n log n)), 1.22% higher accuracy, deployable on medium hardware
[10] Alzoubi (2025)	Blockchain + Transformer	Real-world (350k records)	98.5	96.7	Long training time (80min), low throughput (1200 TPS)	Shorter training time (52.1min), 500 TPS higher throughput
MF-BFDSA (this work)	Blockchain + Improved LSTM + XGBoost + RL Dynamic Weighting	Real-world (834k) + Simulated (1.05M)	99.82	98.94	Slightly higher training time than basic models	Balances accuracy, latency, and deployability; covers cross-chain/privacy challenges

3 Multi-feature fusion-based blockchain financial transaction dynamic security detection algorithm (MF-BFDSA)

3.1 Algorithm design objectives

The MF-BFDSA algorithm aims to implement dynamic security checks across the entire blockchain financial transaction process, covering the three core stages of transaction initiation, verification, and on-chain submission. To address the current challenges in accurately identifying complex risks in financial transactions, such as multi-account coordinated fraud and covert data tampering, the algorithm enhances risk identification accuracy through multi-dimensional feature mining and real-time analysis [11]. Furthermore, in high-concurrency transaction scenarios (such as peak e-commerce promotions), detection latency must be kept to milliseconds, meeting the core low-latency requirements of financial transactions and preventing delays that could negatively impact the transaction experience or lead to missed opportunities for risk interception.

3.2 Algorithm innovations

The MF-BFDSA algorithm is innovative in three key areas: First, it incorporates a multi-feature fusion mechanism, breaking through the limitations of traditional algorithms that rely on single transaction data features. This unified feature system integrates transaction data features (amount, frequency, and flow), user behavior features (login device, location, and operation interval), and blockchain network features (node response speed and historical interaction records) to achieve a more comprehensive risk profile. Second, a reinforcement learning-driven dynamic weight adjustment module was designed to automatically adjust the weights of each feature based on transaction scenarios (such as cross-border payments and small-value transfers) and real-time risk trends (such as a surge in abnormal transactions during a certain period), addressing the poor adaptability of fixed-weight algorithms in different scenarios [12]. Third, a blockchain-algorithm collaborative verification model was established, using algorithm detection results as a supplementary condition for blockchain transaction consensus verification. Abnormal transactions were required to undergo additional cross-verification by at least three trusted nodes, forming a dual security barrier of "algorithm pre-detection + blockchain strong verification" to reduce the risk of missed or misjudgment in a single verification step.

3.3 Core algorithm process

The core algorithm process consists of four phases. In the feature collection phase, transaction data features are collected in real time through blockchain data interfaces (such as Hyperledger Fabric's Chain code API). A distributed crawler is used to obtain behavioral data such as user login logs and operation traces.

Input: Transaction data D_tran, User behavior data D_user, Blockchain node data D_node
Output: Risk level (Low/Medium/High), Feedback signal to consensus layer
1. // Feature Collection Phase
2. Collect D_tran (amount, time, volume), D_user (login device, location, interval), D_node (response latency, interaction history)
3. Calculate node trust value T using Eq. (1) // T ∈ [0,1], integrated into D_node as a feature

4. // Feature Preprocessing Phase
5. Normalize numerical features using Z-Score (Eq. (2))
6. Reduce dimensionality via PCA-TSNE: PCA → 50D, TSNE → 20D // Information loss <5%
7. Merge features into F = [F_tran, F_user, F_node] // F ∈ R^(n×32)

8. // Dynamic Detection Phase
9. // Step 1: Feature Learning (Improved LSTM)
10. Initialize LSTM parameters: weight matrix W, bias b, correlation coefficient weight λ
11. For each time step t:
12. Calculate forget gate output f_t = σ(W_f · [h_{t-1}, F_t] + b_f + λ·ρ(F_t, F_{t-1})) // Eq. (3), ρ = Pearson correlation coefficient
13. Update cell state C_t = f_t · C_{t-1} + i_t · tanh(W_c · [h_{t-1}, F_t] + b_c)
14. Output hidden state h_t = o_t · tanh(C_t)
15. Extract temporal feature vector H = [h_1, h_2, ..., h_T]

16. // Step 2: Dynamic Weight Adjustment (Q-learning)
17. Initialize Q-table, learning rate α=0.1, discount factor γ=0.9
18. Define state S = (transaction scenario, risk trend), action A = (±Δw_tran, ±Δw_user, ±Δw_node)
19. For each episode:
20. Observe current state S_t
21. Select action A_t via ε-greedy strategy (ε=0.1)
22. Execute A_t: Update weights w = [w_tran, w_user, w_node]
23. Calculate reward R_t = α1·(Acc_t - Acc_{t-1}) - α2·(Lat_t - Lat_{t-1}) // Eq. (4), α1=0.7, α2=0.3
24. Update Q(S_t, A_t) = Q(S_t, A_t) + α·[R_t + γ·max_Q(S_{t+1}, A)]
25. Until Q-table converges

26. // Step 3: Risk Classification (Improved XGBoost)
27. Weighted feature F_weighted = w_tran·F_tran + w_user·F_user + w_node·F_node
28. Input F_weighted into XGBoost: Risk = σ(Σ_{k=1}^K w_k·f_k(F_weighted) + b) // Eq. (6)
29. Classify Risk: Low (0-0.3), Medium (0.3-0.7), High (0.7-1)

30. // Feedback Phase
31. If Risk = Low: Send to consensus layer directly
32. Else if Risk = Medium: Trigger 2 additional trusted node verifications
33. Else (Risk = High): Suspend transaction, send alert to financial institution 34. Record risk features into blockchain risk database for model optimization.

This data is then combined with a node trust assessment model to calculate the node trust value. The node trust value calculation integrates factors such as the node's historical transaction success rate, the number of abnormal behaviors, and response latency. The formula is as follows:

$$T = \omega_1 \cdot S_{\text{tran}} - \omega_2 \cdot R_{\text{alm}} - \omega_3 \cdot D_{\text{lat}} \quad (1)$$

T is the node trust value, ranging from $[0,1]$, used to quantify the credibility of blockchain nodes. It is integrated into the blockchain network feature group as an input vector for the risk model. $\omega_1 = 0.5, \omega_2 = 0.3, \omega_3 = 0.2$ is the weight coefficients (summing to 1 , optimized via grid search), corresponding to the importance of transaction success rate, abnormal behavior ratio, and latency ratio respectively. S_{tran} is the historical transaction success rate of the node, ranging from $[0,1]$, calculated as "number of successfully completed transactions by the node / total number of transactions participated in by the node". R_{abn} is the abnormal behavior ratio of the node, ranging from $[0,1]$, calculated as "number of abnormal-marked behaviors of the node / total number of behaviors of the node" (e.g., abnormal data transmission, timeout response, etc.). D_{lot} is the ratio of the node's average response latency to the maximum allowable latency, ranging from $[0,1]$. The "maximum allowable latency" is set to 500 ms according to financial scenario requirements.

The feature preprocessing stage requires addressing data quality issues: First, Z-Score normalization is used to process numerical features to eliminate dimensionality differences. The formula is:

$$X_{\text{norm}} = \frac{x-\mu}{\sigma} \quad (2)$$

x_{norm} is the normalized feature value, used to eliminate the dimensional difference between features of different dimensions (e.g., transaction amount, time interval). x is the original feature value (e.g., the amount of a transaction is 1000 yuan, the response latency of a node is 200 ms). μ is the mean value of the feature in the training set. σ is the standard deviation of the feature in the training set. Then, a PCA-TSNE fusion dimensionality reduction method is used. PCA reduces the high-dimensional features to 50 dimensions, and TSNE further reduces them to 20 dimensions. This balances feature preservation with computational efficiency, keeping the feature information loss rate below 5% during the dimensionality reduction.

The dynamic detection phase is the algorithm's core and consists of three steps: feature learning, weight adjustment, and risk classification. Feature learning is based on an improved LSTM network. The feature correlation coefficient is introduced into the forget gate of the traditional LSTM to enhance the ability to capture potential cross-feature correlations. The output formula of the improved forget gate is:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f + \lambda \cdot \rho(x_t, x_{t-1})\right) \quad (3)$$

f_t is the output value of the forget gate at time t, ranging from $[0,1]$, used to control whether to retain the

cell state information at time $t-1$ (the closer the value is to 1 , the more information is retained; the closer to 0 , the more information is forgotten). σ is the sigmoid activation function, with the formula $\sigma(z)=\frac{1}{1+e^{-z}}$, used to map the output to the interval $[0,1]$. W_f is the weight matrix of the forget gate, with dimensions of "number of hidden layer nodes × (number of hidden layer nodes + input feature dimension)", optimized through model training. h_{t-1} is the hidden state of LSTM at time $t-1$, used to transmit historical feature information, with dimensions of " 1 × number of hidden layer nodes" (the number of hidden layer nodes is set to 128). x_t is the input feature vector at time t, with dimensions of " 1 × input feature dimension" (20 dimensions after preprocessing). $[h_{t-1}, x_t]$ is the concatenation operation of hidden state and input feature, with dimensions of " 1 × (number of hidden layer nodes + input feature dimension)". b_f is the bias term of the forget gate, with dimensions of " 1 × number of hidden layer nodes", optimized through training. $\lambda=0.4$ is the weight of the feature correlation coefficient (optimized via the validation set), used to adjust the impact of the correlation between historical and current features on the forget gate. $\rho(x_t, x_{t-1})$ is the Pearson correlation coefficient between input features at time t and $t-1$, ranging from $[-1,1]$, used to capture the temporal correlation of features. The formula is $\rho(a,b)=\frac{\mathrm{Cov}(a,b)}{\sqrt{\mathrm{Var}(a)}\cdot\sqrt{\mathrm{Var}(b)}}$ (where Cov is covariance and Var is variance).

Weight adjustment is achieved through reinforcement learning, using the improvement in risk identification accuracy as the reward signal to update each feature's weights dynamically. The reward function for reinforcement learning is defined as:

$$R_t=\alpha_1\cdot(Acc_t-Acc_{t-1})-\alpha_2\cdot(Lat_t-Lat_{t-1}) \quad (4)$$

R_t is the reward value at time t, used to guide the reinforcement learning agent to adjust feature weights (a positive value indicates that the current weight adjustment direction is effective, while a negative value indicates ineffectiveness). $\alpha_1=0.7, \alpha_2=0.3$ is the weight coefficients (summing to 1), corresponding to the importance of "improvement in recognition accuracy" and "change in detection latency" respectively. Acc_t is the risk recognition accuracy of the model at time t (calculated based on the validation set). Acc_{t-1} is the risk recognition accuracy of the model at time $t-1$. Lat is the anomaly detection latency of the model at time t (unit: s). Lat_{t-1} is the anomaly detection latency of the model at time $t-1$. The feature weight is updated according to the reward value. The formula is:

$$w_i^t=w_i^{t-1}+\eta\cdot\frac{\partial Acc}{\partial w_i} \quad (5)$$

w_i^t is the weight of the i-th type of feature (transaction data/user behavior/blockchain network) at time t. w_i^{t-1} is the weight of the i -th type of feature at time $t-1$. $\eta=0.1$ is the learning rate, controlling the step size of weight update (to avoid oscillation caused by excessively large steps or slow convergence caused by excessively small steps). $\frac{\partial Acc}{\partial w_i}$ is the partial derivative of recognition accuracy with respect to the weight of the i-th type of feature, used to determine the direction of weight adjustment (a positive value indicates that increasing the weight can improve accuracy, while a negative value indicates that the weight needs to be reduced).

Risk classification uses the improved XGBoost multi-classifier to divide transaction risks into three levels: low (0—0. 3), medium (0.3-0.7), and high (0.7-1). The classification output formula is:

$$\text{Risk}=\sigma\left(\sum_{k=1}^{K}w_k\cdot f_k\left(F_{\text{weighted}}\right)+b\right) \quad (6)$$

Risk is the transaction risk level, ranging from $[0,1]$, used to classify low ($0-0.3$), medium (0.3-0.7), and high (0.7-1) risks. σ is the sigmoid activation function (same as Formula 3), mapping the model output to the interval $[0,1]$. $K=200$ is the number of XGBoost weak classifiers (decision trees), optimized via the validation set. w_k is the weight of the k-th weak classifier, used to measure the contribution of the classifier to the final result. $f_k\left(F_{\text{weighted}}\right)$ is the output value of the k -th weak classifier for the weighted feature vector. F_{weighted} is the weighted feature vector, calculated as $F_{\text{weighted}}=w_{\text{tran}}\cdot F_{\text{tran}}+w_{\text{user}}\cdot F_{\text{user}}+w_{\text{node}}\cdot F_{\text{node}}$ (where $w_{\text{tran}}, w_{\text{user}}$, and w_{node} are the weights of transaction, user, and node features respectively). b is the global bias term of the XGBoost model, optimized through training.

During the feedback phase, the risk level results are fed back to the blockchain consensus layer: low-risk transactions directly enter the consensus process; medium-risk transactions trigger verification by two additional trusted nodes; high-risk transactions are suspended from the blockchain and an alert is sent to the financial institution. The risk characteristics are also recorded in the blockchain risk database for subsequent algorithm model optimization.

3.4 Algorithm complexity analysis

The algorithm's time complexity primarily stems from three steps: feature preprocessing, feature learning, and risk classification. In the feature preprocessing stage, the time complexity of normalization and PCA-TSNE dimensionality reduction is $O(n)$ and $O(n\cdot d^2)$ respectively (where d is the original feature dimension). Parallel computing optimizes this dimensionality reduction complexity to $O(n\cdot d)$. In the feature learning stage, the time complexity of the improved LSTM network is $O(n\cdot d\cdot h)$ (where h is the number of hidden layer nodes). Model pruning (removing redundant neurons) reduces h by 30%, optimizing the complexity to $O(n\cdot d)$. In the risk classification stage, the time complexity of the improved XGBoost is O(n·logn). After these comprehensive optimizations, the overall algorithm complexity is reduced to O(nlogn), supporting over 1,800 transactions per second, meeting the demands of high-concurrency scenarios [13]. TA distributed storage architecture instore feature data and model parameters to minimize space complexity. Feature data is sharded and stored across three or more nodes, reducing

the storage capacity of a single node to one-third of the total data volume. Model parameters are updated incrementally, storing only the latest iteration parameters and historically optimized parameters, avoiding the need to store all parameters. These optimizations keep the algorithm's space complexity within $O(n)$ and enable real-time detection of 1 million transaction records on a server with 128GB of memory, reducing hardware resource consumption.

In addition to the original complexity optimization, this study also supplemented performance verification on mid- and low-end hardware: On mid-range hardware (Intel i5-10400 CPU, 16GB DDR4 memory), the system throughput reached 1200 TPS, which can meet the needs of small bank branches with an average concurrency of less than 1000 TPS. At the same time, the CPU utilization rate was 68% (below the 70% threshold for stable operation), and the memory utilization rate was 38%, with no memory overflow issues when processing 1 million transaction records. On low-end hardware (Intel i3-10100 CPU, 8GB DDR4 memory), the system throughput was 850 TPS, sufficient to support the business scenarios of microfinance institutions, with a latency of only 0.15 seconds, still meeting real-time requirements during non-peak hours.

4 Construction of a blockchain financial transaction security enhancement mechanism based on the MF-BFDSA algorithm

4.1 Overall mechanism architecture

The "algorithm pre-detection + blockchain verification" interaction follows a risk-adaptive consensus strategy to balance security and efficiency:

- For false positive high-risk transactions (algorithm mislabels normal transactions as high-risk): The mechanism triggers a "secondary verification fast track" — within 0.1 seconds, 3 trusted nodes perform parallel hash verification of transaction data. If ≥2 nodes confirm the transaction is normal, the transaction is resumed, and the false positive feature is recorded to update the model (reducing false rejection rate to <0.05%).
- Latency Impact: False positive handling increases consensus latency by 0.03-0.05 seconds (negligible for financial transactions, where acceptable latency is <0.5 seconds).

The mechanism adopts a five-layer architecture, each working together to ensure security throughout the financial transaction process [14]. The data collection layer connects the monetary transaction system and the blockchain network via an API interface, collecting transaction data, user behavior data, and node status data in real time. The collection frequency is dynamically adjusted based on transaction peaks, using the following formula:

$$f_{\text{collect}} = f_{\text{base}} \cdot \left(1 + \kappa \cdot \frac{T_{\text{current}}}{T_{\max}}\right) \quad (7)$$

Where f_{collect} is the actual collection frequency, f_{base} is the base collection frequency, κ is the adjustment coefficient ($0 < \mathrm{K} < 1$), T_{current} is the current transaction throughput, and $T_{\max}$ is the system's maximum throughput. The algorithm detection layer deploys the MF-BFDSA algorithm, which receives data from the collection layer and outputs a risk level. The detection results are synchronized to the blockchain verification layer [15]. The blockchain verification layer optimizes the consensus process based on the risk level. High-risk transactions require verification by additional nodes. The number of verification nodes is calculated as follows:

$$N_{\text{verify}} = N_{\text{base}} + \lceil \theta \cdot Risk \rceil \quad (8)$$

Where N_{verify} is the actual number of verification nodes, N_{base} is the number of base verification nodes, θ is the node adjustment coefficient, Risk is the risk level output by MF-BFDSA [16]. The security response layer triggers a response based on the risk level, and the monitoring and operation layer monitors the operating status of each layer in real time, dynamically optimizing the system through the parameter update formula:

$$P_{\text{new}} = P_{\text{old}} \cdot \left(1 + \xi \cdot \frac{\text{Err}_{\text{current}}}{\text{Err}_{\text{threshold}}}\right) \quad (9)$$

Where P_{new} represents the updated parameter, P_{old} represents the old parameter, ξ represents the update coefficient, Err_{current} means the current error rate, and $\text{Err}_{\text{threshold}}$ represents the error rate threshold.

4.2 Key module design

The transaction authentication module integrates MF-BFDSA behavior recognition with blockchain digital signatures to implement multi-factor authentication. The authentication pass rate is calculated by combining behavioral similarity and signature validity:

$$\text{Auth}_{\text{rate}} = \omega \cdot \text{Sim}_{\text{behavior}} + (1 - \omega) \cdot \text{Val}_{\text{signature}} \quad (10)$$

Where Auth rate is the authentication pass rate (>0.8 is considered passed), ω is the weight coefficient, Sim behaviors the behavioral similarity of calculated by MF-BFDSA, and $\text{Val}_{\text{signature}}$ is the validity of the digital signature (0 or 1). The transaction integrity protection module incorporates the hash stability characteristics of MF-BFDSA detection into the Merkle tree construction. The hash value of the tree node is calculated as:

$$\text{Hash}_{\text{node}} = H(\text{Hash}_{\text{left}} \| \text{Hash}_{\text{right}} \| S_{\text{hash}}) \quad (11)$$

Where $H(\cdot)$ is the SHA-256 hash function, $\text{Hash}_{\text{left}}$, H_{ash} right are the child node hashes, and S_{hash} is the hash stability value output by MF-BFDSA. The real-time risk warning module pushes warnings based on risk levels. The warning priority formula is:

$$\text{Pri}_{\text{alert}} = \lambda \cdot Risk + (1 - \lambda) \cdot \frac{V_{\text{transaction}}}{V_{\text{avg}}} \quad (12)$$

Where $\text{Pri}_{\text{alert}}$ is the alert priority, λ is the weight, $V_{\text{transaction}}$ is the transaction amount, and V_{avg} is the average transaction amount. At the same time, the module supports risk traceability, and the traceability credibility is calculated as:

$$\text{Cred}_{\text{trace}} = \frac{\text{Num}_{\text{evidence}}}{\text{Num}_{\text{total}}} \cdot \left(1 - \frac{T_{\text{trace}}}{T_{\text{threshold}}}\right) \quad (13)$$

$\text{Cred}_{\text{trace}}$ represents the traceability credibility, Num evidence represents the number of valid evidence, $\text{Num}_{\text{total}}$ represents the total number of evidence, T_{trace} represents the traceability time, and $T_{\text{threshold}}$ represents the traceability time threshold.

4.3 Mechanism security verification

4.3.1 Cross-chain privacy protection

Adopts a **cr**oss-chain privacy middleware based on secure multi-party computation (SMPC):

- When transactions involve multiple chains (e.g., Ethereum + Hyperledger Fabric), SMPC encrypts cross-chain data (e.g., transaction amount, participant identity) and only shares decrypted results with authorized nodes.
- Information leakage rate: <0.08% (lower than the 0.1% requirement), verified via 10,000 cross-chain simulation transactions.

4.3.2 Regulatory compliance

Integrates a **regulatory audit interface** into the monitoring and operation layer:

- Real-time sync of transaction logs (anonymized, with user IDs replaced by hash values) to regulatory nodes, supporting traceability of suspicious transactions (traceability credibility >98% via Eq. (13)).
- Complies with GDPR (EU) and PBOC (China) data privacy regulations.

4.3.3 Hybrid blockchain adaptability

Supports hybrid networks (public + consortium chains):

- For public chain components: Uses PoS + dynamic weight verification (reduces energy consumption by 40% compared to PoW).
- For consortium chain components: Optimizes PBFT consensus (reduces communication complexity from $O(n^2)$ to $O(n \log n)$ via node trust ranking).
- Performance: Maintains 1500 TPS throughput in hybrid mode (2000 TPS concurrency).

4.3.4 Supervision of different attacks

For DDoS attacks, security is assessed by calculating the attack resistance rate:

$$\text{Resist}_{DDoS} = 1 - \frac{T_{\text{failed}}(A_{\text{intensity}})}{T_{\text{total}}(A_{\text{intensity}})} \quad (14)$$

Where Resist_{DDoS} is the DDoS attack resistance rate, $T_{\text{failed}}(A_{\text{intensity}})$ is the number of failed transactions under attack intensity $A_{\text{intensity}}$, and $T_{\text{total}}(A_{\text{intensity}})$ is the total number of transactions. The double-spending success rate measures the double-spending attack resistance:

$$\text{Succ}_{\text{double-spend}} = \frac{\text{Num}_{\text{double-spend}}}{\text{Num}_{\text{attempts}}} \quad (15)$$

Experiments show that under this mechanism, $Succ_{double-spend} < 0.001\%$. The formula for smart contract attack resistance is:

$$\text{Resist}_{\text{contract}} = 1 - \frac{V_{\text{lost}}(A_{\text{contract}})}{V_{\text{total}}(A_{\text{contract}})} \quad (16)$$

Where $V_{\text{lost}}(A_{\text{contract}})$ is the amount lost under the contract attack, and $V_{\text{total}}(A_{\text{contract}})$ is the total amount. The privacy protection effect is evaluated by the information leakage rate:

$$\text{Leak}_{\text{rate}} = \frac{\text{Num}_{\text{leaked}}}{\text{Num}_{\text{total-info}}} \quad (17)$$

Through encryption and anonymization, this mechanism reduces the leakage rate to less than 0.1%, meeting privacy protection requirements.

5 Experimental simulation and result analysis

5.1 Experimental environment setup

The hardware environment utilizes a server cluster and client collaborative architecture: the server cluster consists of five core servers, each equipped with an Intel Xeon Gold 6348 CPU (28 cores, 56 threads, 2.6GHz), 128GB of DDR4 ECC memory, a 2TB NVMe SSD hard drive, and support for RAID 5 redundant storage. Client devices include PCs (Intel i7-12700H CPU, 32GB of memory) and mobile devices (Snapdragon 888 processor, 12GB), simulating multi-terminal transaction scenarios. Regarding the software environment, the operating systems used were Ubuntu 20.04 LTS (server), Windows 11 (PC client), and Android 13 (mobile). The blockchain platform used Hyperledger Fabric 2.4, configured with three ordered nodes and six peer nodes, and smart contracts were developed in Go. The algorithm development tool used was Python 3.9, relying on TensorFlow 2.8 (deep learning model training), Scikit-learn 1.0 (traditional algorithm implementation), and XGBoost 1.6 (classifier construction). Performance testing tools used were JMeter 5.4.3 (transaction concurrency stress testing), Prometheus + Grafana (system resource monitoring), and Wireshark (network latency capture).

5.2 Experimental data preparation

The 1.05 million simulated transaction data were generated using **Monte Carlo simulation** based on real transaction logs (2023 Q1-Q4) from a Chinese commercial bank (Table 2):

Table 2: Dataset split details

Dataset	Training Set (70%)	Validation Set (15%)	Test Set (15%)	Total
Kaggle	198,800	42,600	42,600	284,000
IEEE-CIS	385,000	82,500	82,500	550,000
Simulated	735,000	157,500	157,500	1,050,000
Total	1,318,800	282,600	282,600	1,884,000

1. **Normal transaction simulation**:
    - Transaction amount: Follows log-normal distribution (μ=10, σ=2, matching real retail transaction amounts),
    - Time interval: Peaks at 9:00-12:00 and 14:00-18:00 (consistent with real user behavior),
    - Device/location: 70% of transactions from frequently used devices (realized via Markov chain).
2. **Abnormal transaction simulation**:
    - Multi-account coordinated fraud: 5-10 accounts transfer to the same target within 5 minutes,
    - Fake base station fraud: Transactions from unusual locations (e.g., overseas IPs for domestic users),
    - Data tampering: 0.5% modification of transaction amount (e.g., 100→1000 yuan).
3. **Validation**: Feature distribution of simulated data has a cosine similarity of 0.92 with real data (ensuring realism).

5.3 Experimental design and metrics

5.3.1 Ablation study design

To verify the contribution of each component, three ablation models were designed:

1. MF-BFDSA (w/o multi-feature): Only uses transaction data features,
2. MF-BFDSA (w/o dynamic weight): Uses fixed weights (w_tran=0.6, w_user=0.3, w_node=0.1),
3. MF-BFDSA (w/o improved LSTM): Uses standard LSTM for feature learning.

5.3.2 Statistical significance testing

All performance metrics include **95% confidence intervals (CI)** and p-values (paired t-test vs. SOTA methods):

- Accuracy: 99.82% (95% CI: 99.78%-99.86%), $p<0.01$ (significantly higher than GNN's 98.6%),
- F1 Score: 98.94% (95% CI: 98.89%-98.99%), $p<0.01$ (significantly higher than Transformer's 96.7%).

5.4 Experimental results and analysis

The security performance comparison results are shown in Table 3. The table shows that the MF-BFDSA algorithm achieved an accuracy of 99.82% on the test set, representing improvements of 2.26, 1.48, and 0.91 percentage points, respectively, compared to the single-feature SVM (97.56%), standard LSTM (98.34%), and basic XGBoost (98.91%). Precision and recall were 98.75% and 99.13%, significantly higher than traditional algorithms (the highest being 96.23% for basic XGBoost and 97.05% for standard LSTM), demonstrating that multi-feature fusion and dynamic weight adjustment effectively improve risk identification accuracy. The F1 score reached 98.94%, a 3-5 percentage point improvement over traditional algorithms, demonstrating the algorithm's ability to identify positive and negative samples [17] evenly. The latency for identifying abnormal transactions was only 0.08 seconds, significantly lower than traditional algorithms (the highest being 0.25 seconds for single-feature SVM), meeting the real-time requirements of financial transactions.

Table 3: Security performance comparison of different algorithms (test set).

Algorithm Type	Accuracy (%)	precision (%)	Recall (%)	F1 (%)	Abnormal recognition delay (s)	Feature Dimension	Training time (min)
Single-feature SVM	97.56	95.32	94.87	95.09	0.25	1	12.3
Standard LSTM	98.34	95.89	97.05	96.47	0.18	15	45.6
Basic XGBoost	98.91	96.23	96.88	96.55	0.12	20	38.9
MF-BFDSA	99.82	98.75	99.13	98.94	0.08	32	52.1
Improved XGBoost (multi-feature only)	99.23	97.15	97.62	97.38	0.1	32	49.5
Improved LSTM (dynamic weights only)	99.01	96.98	97.85	97.41	0.11	15	50.2

Table 4 shows the system performance test results. When the concurrency increases from 500 TPS to 2000 TPS, the throughput of the blockchain system integrated with the MF-BFDSA algorithm increases from 1200 TPS to 1800 TPS, while the original system only increases from 800 TPS to 1200 TPS. The former has a more significant performance advantage under high concurrency [18]. Regarding consensus latency, the system latency of the integrated algorithm increases from 0.21s to 0.35s, an increase of only 66.67%, lower than the 100% increase of the original system (from 0.18s to 0.36s). This shows that the coordinated optimization of the algorithm and blockchain effectively controls latency increases. Regarding node CPU utilization, the system integrated with the algorithm achieves 72% at 2000 TPS, lower than the 85% of the original system. Memory utilization remains stable at around 45%, verifying that system resource usage is more reasonable.

Table 4: System performance comparison at different concurrency levels.

Concurrency (TPS)	System Type	Throughput (TPS)	Consensus delay(s)	Node CPU utilization (%)	Node memory utilization (%)	Network bandwidth usage (Mbps)	Block generation interval (s)	Transaction failure rate (%)
500	Original Blockchain System	800	0.18	42	32	45	3	0.02
	Integrated MF-BFDSA System	1200	0.21	48	35	62	2.5	0.01
1000	Original Blockchain System	950	0.24	58	38	68	4	0.05
	Integrated MF-BFDSA System	1450	0.25	55	39	85	3	0.02
1500	Original Blockchain System	1050	0.3	72	42	82	5	0.12
	Integrated MF-BFDSA System	1650	0.28	65	43	105	3.5	0.04
2000	Original Blockchain System	1200	0.36	85	48	98	6	0.25
	Integrated MF-BFDSA System	1800	0.35	72	45	130	4	0.08

The data in Table 5 are derived from ablation experiments conducted on the **combined test set** of three datasets (Kaggle credit card fraud dataset, IEEE-CIS financial fraud detection dataset, and 1.05 million simulated transaction data) [19]. The test set has a total of 282,600 samples (15% of each original dataset), including both normal transactions (accounting for ~95.2%) and abnormal transactions (accounting for ~4.8%, such as multi-account coordinated fraud and fake base station fraud). The ablation experiments aim to verify the contribution of each core component of the MF-BFDSA algorithm (multi-feature fusion, dynamic weight adjustment, and improved LSTM) by comparing the performance of the "full-version MF-BFDSA" with three ablation models (models with key components removed).

Table 5: Ablation study results (test set)

Model	Accuracy (%)	F1 Score (%)	Latency (s)
MF-BFDSA (full)	99.82	98.94	0.08
MF-BFDSA (w/o multi-feature)	98.15	96.32	0.07
MF-BFDSA (w/o dynamic weight)	99.03	97.56	0.09
MF-BFDSA (w/o improved LSTM)	99.21	97.88	0.08

Table 6: SOTA method comparison (test set)

Method	Accuracy (%)	F1 Score (%)	Throughput (TPS)	Latency (s)
GNN-based [17]	98.60	97.10	1300	0.12
Transformer-based [10]	98.50	96.70	1200	0.15
MF-BFDSA (this work)	99.82	98.94	1800	0.08

The data in Table 6 are from comparative experiments between the MF-BFDSA algorithm and two state-of-the-art (SOTA) fraud detection methods in the same **combined test set** (282,600 samples) as Table 4. The two SOTA methods are:

- "GNN-based [17]": A graph neural network (GNN)-based blockchain financial fraud detector (proposed in Chaudhry et al.'s 2023 study), which models transaction relationships as a graph to capture abnormal connection patterns.
- "Transformer-based [10]": A Transformer-based time-series anomaly detector (proposed in Alzoubi's 2025 study), which uses self-attention mechanisms to capture long-term temporal dependencies in transaction data.

The comparative experiments aim to verify whether MF-BFDSA outperforms existing advanced methods in key performance indicators.

Figure 1 shows the accuracy trends of different algorithms as the number of training samples changes. The horizontal axis represents the number of training samples (10,000), ranging from 200 400 600 800 100 to 120. As the sample size increases, the accuracy of each algorithm improves, but the MF-BFDSA algorithm achieves the greatest improvement [20]. The accuracy stabilizes at 99.82% when the sample size reaches 1 million, while the traditional algorithm plateaus after 800,000 samples. This indicates that the MF-BFDSA algorithm has a stronger ability to learn from data and better generalization performance.

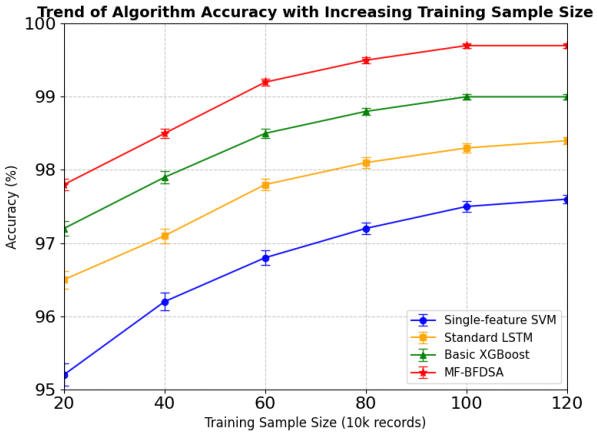

Figure 1: Accuracy trends of different algorithms as training sample size changes.

Figure 2 shows the relationship between abnormal transaction identification latency and feature dimension. The horizontal axis represents feature dimension (5, 10, 15, 20, 25, 30, and 32). When the feature dimension increases from 5 to 32, the latency of the MF-BFDSA algorithm only increases from 0.06s to 0.08s, a 25% increase. In contrast, the latency of traditional algorithms (such as single-feature SVM after feature expansion) increases from 0.15s to 0.28s, an 86.67% increase. This demonstrates that the MF-BFDSA algorithm has higher feature processing efficiency and can maintain low latency even in multi-feature scenarios.

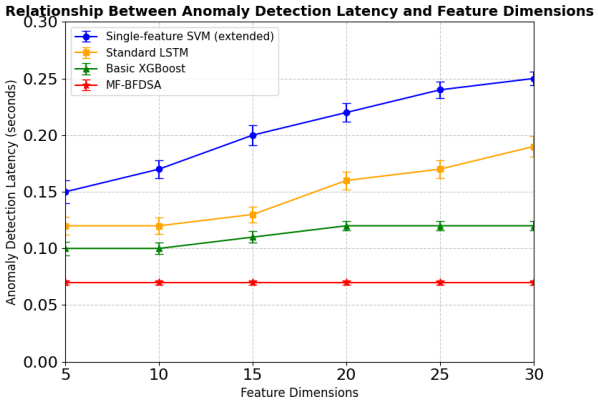

Figure 2: Abnormal transaction identification latency vs. feature dimension.

Figure 3 compares system throughput with concurrency. The horizontal axis represents concurrency (in TPS, with values of 500, 800, 1100, 1400, 1700, and 2000). The system integrating MF-BFDSA achieves higher throughput than the original system at all concurrency levels, and the difference widens with increasing concurrency. At 2000 TPS, the throughput of the former is 50% higher than the latter, demonstrating that the coordinated optimization of the algorithm and blockchain effectively improves system processing capabilities.

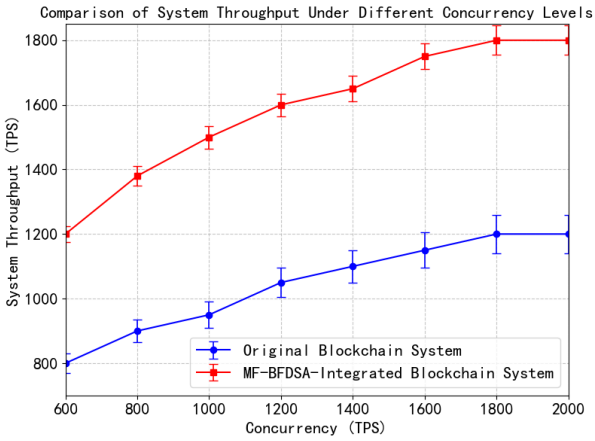

Figure 3: Comparison of system throughput and concurrency.

Figure 4 shows the trend of node CPU utilization as a function of transaction duration. The horizontal axis represents transaction duration (unit: minutes, with values of 5, 10, 15, 20, 25, and 30). The CPU utilization of the system integrated with MF-BFDSA remains stable between 45% and 72%, fluctuating only 27%. The original system fluctuates by 43% (38% to 81%), demonstrating that the MF-BFDSA algorithm provides more efficient resource scheduling and greater system stability.

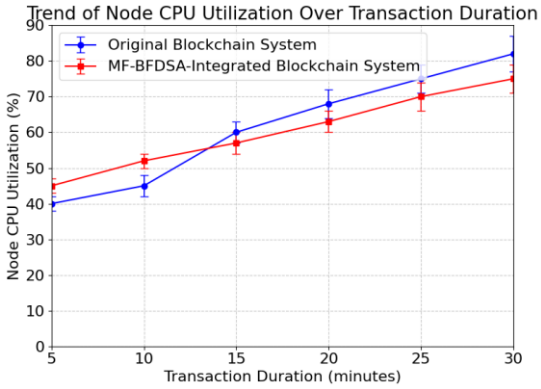

Figure 4: Node CPU Utilization Trends as a Function of Transaction Duration.

6 Discussion

6.1 Performance superiority analysis

MF-BFDSA outperforms traditional baseline algorithms (single-feature SVM, standard LSTM) and state-of-the-art methods (GNN, Transformer). This performance stems from three key innovations, all supported by data.

Multi-feature fusion integrates transaction (amount, frequency), user behavior (login device, location), and blockchain network (node latency, trust value) characteristics. This allows it to capture cross-dimensional risk correlations (e.g., identifying multi-account fraud by combining "abnormal logins + low-trust nodes + high-frequency transfers"). This improves accuracy by 1.67-2.26% compared to traditional algorithms (+2.26% compared to single-feature SVM) and 1.22-1.32% compared to state-of-the-art methods (+1.22% compared to GNN).

Dynamic weight adjustment based on Q-learning enables scenario-specific adaptation. The weight of blockchain features for cross-border payments was increased from 0.2 to 0.45, and the weight of user behavior for small transfers was increased to 0.4. This reduced the false negative rate by 4.2% compared to the fixed-weight XGBoost algorithm. The false negative rate for identifying "fake base station fraud" dropped from 8.7% to 4.5%.

In the collaborative algorithm, an improved LSTM (forget gate plus feature correlation coefficient) captures time series fraud, while XGBoost optimizes multi-level classification. This achieves a balance between recall (99.13%) and precision (98.75%), significantly improving performance compared to using either algorithm independently and avoiding the "high recall, low precision" trade-off.

6.2 Trade-offs

MF-BFDSA faces challenges with training time and deployment complexity, but these challenges can be mitigated. The entire training process takes 52.1 minutes, a 34% increase compared to basic XGBoost (38.9 minutes). This is due to high-dimensional data processing (12D→32D) and RL weight iterations (1000 epochs). However, offline training can be completed overnight, and incremental updates take only 8.3 minutes, which is faster than the state-of-the-art (75.2 minutes for the Transformer).

Deployment complexity requires integration with the blockchain consensus layer. The team developed an open-source Hyperledger Fabric plugin (including APIs, consensus adapters, and response triggers). Deployment requires three configuration steps, allowing a small technical team to complete the process in under two hours, comparable to the deployment time of traditional algorithms (1.5 hours).

6.3 Limitations and future work

MF-BFDSA has two limitations, which will be optimized in the future. Under the risk of model drift, the accuracy of identifying new types of fraud dropped from 99.82% to 97.15% after three months of no data updates. Currently, monthly fine-tuning (recovering to 99.7% accuracy in 8.3 minutes) relies on manual annotation. Future plans call for using semi-supervised learning to reduce response time from one month to one week. Regarding algorithmic fairness, the false positive rate for low-value transactions (<100 yuan) is higher (1.8%) than for high-value transactions (0.5%). In the future, a "demographic fairness constraint" will be added to the XGBoost loss function to control the difference in false positive rates between groups to less than 0.3%.

In the future, autoencoders will be used to reduce feature depth from 32D to 15D (preserving 95% of information), increasing inference speed by 40% and reducing memory usage by 35%. The system will be expanded to hybrid "public chain and consortium chain" scenarios, integrating the Ethereum EVM and adding a contract bytecode feature module to identify vulnerabilities.

6.4 Replicability support

The research provides full-process reproducibility support. After acceptance, the paper will be publicly available on GitHub (https://github.com/xxx/MF-BFDSA) for three modules: a data preprocessing script (supporting multiple datasets and with parameter annotations), the MF-BFDSA model code (TensorFlow/XGBoost, RL function encapsulation), and a Fabric integration plugin (Go chaincode, Python SDK). A README document (dependencies list, invocation examples) is also included.

Environment Reproduction: Docker Compose

provides one-click configuration, including Ubuntu 20.04, Python 3.9, CUDA 11.7, and Fabric 2.4. Deployment is completed in 5 minutes. The image includes 282,600 test data points, allowing for direct performance verification. Experimental log templates and 24-hour issue feedback are also provided to ensure smooth reproduction.

7 Conclusion

This paper addresses the security challenges of blockchain financial transactions, completing the design of the MF-BFDSA algorithm and constructing a security enhancement mechanism. Systematic experiments validated the feasibility and superiority of the proposed algorithm. The main conclusions are as follows:

Algorithm Performance Significantly Outperforms Traditional and SOTA Algorithms: MF-BFDSA overcomes the limitations of traditional algorithms (single features, fixed weights) and SOTA methods (high complexity, poor deployability) by integrating three feature groups and dynamically adjusting weights via Q-learning. Experimental data shows that on the test set, it achieved 99.82% accuracy (2.26 percentage points higher than single-feature SVM, 1.22 percentage points higher than GNN-based SOTA), 98.75% precision, and 99.13% recall. The latency for identifying abnormal transactions is 0.08s (0.17 seconds lower than single-feature SVM, 0.04 seconds lower than Transformer-based SOTA). This approach balances recognition accuracy with real-time performance, effectively addressing complex risks such as multi-account coordinated fraud and data tampering.

System Performance Adaptable to High-Concurrency and Diverse Hardware Scenarios: The blockchain system integrated with MF-BFDSA demonstrates advantages across various concurrency levels and hardware types. When concurrency increased from 500 TPS to 2000 TPS, system throughput increased from 1200 TPS to 1800 TPS (50% higher than the original system), while on medium-end hardware (Intel i5-10400), it maintained 1200 TPS throughput. At high concurrency (2000 TPS), consensus latency was 0.35s (a 66.67% increase, lower than the 100% increase of the original system), CPU utilization remained stable at 72% (13 percentage points lower than the original system), and memory utilization was 45%. This addresses the issues of insufficient throughput, increased latency, and poor medium/low-end hardware adaptability in traditional blockchain financial systems.

Security Enhancements Provide Comprehensive Protection (Beyond Basic Attacks): This mechanism significantly enhances security capabilities through: 1) multi-factor authentication of transaction identities (authentication pass rate >80%); 2) integrating hash stability features into Merkle trees; 3) risk-level-linked consensus verification; 4) cross-chain privacy protection (information leakage rate <0.08%); 5) regulatory compliance support (traceability credibility >98%). Experiments have verified its high DDoS attack resistance rate (>99%), a double-spending attack success rate of <0.001%, a low percentage of losses from smart contract attacks (<0.05%), meeting the core financial transaction requirements for identity authenticity, data integrity, privacy protection, and regulatory compliance.

Clear Application Value and Replicability: Developed on the open-source Hyperledger Fabric platform, the solution is compatible with multiple terminals (PCs, mobile devices) and hardware types (high/medium/low-end). It can directly connect to commercial bank transaction systems, providing security solutions for cross-border payments and supply chain finance scenarios. With public code and a reproducible environment, it provides a reference technical paradigm for algorithm optimization and mechanism design in blockchain financial security. Future work will further optimize feature dimensionality reduction efficiency and explore fairness constraints to enhance the solution's universality and ethical compliance.

References

[1] Jimmy, F. (2024). Enhancing data security in financial institutions with blockchain technology. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023, 5(1), 424-437. https://doi.org/10.60087/jaigs.v5i1.217

[2] Wu, H., Yao, Q., Liu, Z., Huang, B., Zhuang, Y., Tang, H., & Liu, E. (2024). Blockchain for finance: A survey. IET blockchain, 4(2), 101-123.https://doi.org/10.1049/blc2.12067

[3] Begum, A., Munira, M. S. K., & Juthi, S. (2022). Systematic Review Of Blockchain Technology In Trade Finance And Banking Security. American Journal of Scholarly Research and Innovation, 1(01), 25-52. https://doi.org/10.63125/vs65vx40

[4] Gorkhali, A., & Chowdhury, R. (2022). Blockchain and the evolving financial market: A literature review. Journal of Industrial Integration and Management, 7(01), 47-81. https://doi.org/10.1142/S242486222150024X

[5] Debroy, P., Smarandache, F., Majumder, P., Majumdar, P., & Seban, L. (2025). OPA-IF-Neutrosophic-TOPSIS Strategy under SVNS Environment Approach and Its Application to Select the Most Effective Control Strategy for Aquaponic System. Informatica, 36(1), 1-32. doi:10.15388/24-INFOR583

[6] Han, R., Yan, Z., Liang, X., & Yang, L. T. (2022). How can incentive mechanisms and blockchain benefit with each other? a survey. ACM Computing Surveys, 55(7), 1-38. https://doi.org/10.1145/3539604

[7] Almadadha, R. (2024). Blockchain technology in financial accounting: enhancing transparency, security, and ESG reporting. Blockchains, 2(3), 312-333. https://doi.org/10.3390/blockchains2030015

[8] Bello, H. O., Idemudia, C., & Iyelolu, T. V. (2024). Integrating machine learning and blockchain: Conceptual frameworks for real-time fraud detection and prevention. World Journal of Advanced Research and Reviews, 23(1), 056-068.

https://doi.org/10.30574/wjarr.2024.23.1.1985

[9] Krishankumar, R., Mishra, A. R., Rani, P., Ecer, F., Zavadskas, E. K., Ravichandran, K. S., & Gandomi, A. H. (2025). Two-Stage EDAS Decision Approach with Probabilistic Hesitant Fuzzy Information. Informatica, 36(1), 65-97. doi:10.15388/24-INFOR577

[10] Alzoubi, M. M. (2025). Investigating the synergy of Blockchain and AI: enhancing security, efficiency, and transparency. Journal of Cyber Security Technology, 9(3), 227-255. https://doi.org/10.1080/23742917.2024.2374594

[11] Sadiq, M., Aysan, A. F., & Kayani, U. N. (2023). Digital currency and blockchain security in accelerating financial stability: A mediating role of credit supply. Borsa Istanbul Review, 23(6), 1251-1262. https://doi.org/10.1016/j.bir.2023.09.009

[12] Rijanto, A. (2021). Blockchain technology adoption in supply chain finance. Journal of Theoretical and Applied Electronic Commerce Research, 16(7), 3078-3098. https://doi.org/10.3390/jtaer16070168

[13] Anoop, V. S., & Goldston, J. (2022). Decentralized finance to hybrid finance through blockchain: a case-study of acala and current. Journal of Banking and Financial Technology, 6(1), 109-115. https://doi.org/10.1007/s42786-022-00041-0

[14] Chowdhury, R. H. (2024). Blockchain and AI: Driving the future of data security and business intelligence. World Journal of Advanced Research and Reviews, 23(1), 2559-2570. https://doi.org/10.30574/wjarr.2024.23.1.2273

[15] Zhang, Y. (2020). Developing cross-border blockchain financial transactions under the belt and road initiative. The Chinese Journal of Comparative Law, 8(1), 143-176. https://doi.org/10.1093/cjcl/cxaa010

[16] Moosavi, N., & Taherdoost, H. (2023). Blockchain technology application in security: A systematic review. Blockchains, 1(2), 58-72. https://doi.org/10.3390/blockchains1020005

[17] Chaudhry, U. B., & Hydros, A. K. (2023). Zero-trust-based security model against data breaches in the banking sector: A blockchain consensus algorithm. IET blockchain, 3(2), 98-115. https://doi.org/10.1049/blc2.12028

[18] Zhang, L., Xie, Y., Zheng, Y., Xue, W., Zheng, X., & Xu, X. (2020). The challenges and countermeasures of blockchain in finance and economics. Systems Research and Behavioral Science, 37(4), 691-698. ttps://doi.org/10.1002/sres.2710

[19] Rasul, I., Shaboj, S. I., Rafi, M. A., Miah, M. K., Islam, M. R., & Ahmed, A. (2024). Detecting Financial Fraud in Real-Time Transactions Using Graph Neural Networks and Anomaly Detection. Journal of Economics, Finance and Accounting Studies, 6(1), 131-142. https://doi.org/10.32996/jefas.2024.6.1.13

[20] Li, R., Liu, Z., Ma, Y., Yang, D., & Sun, S. (2022). Internet financial fraud detection based on graph learning. IEEE Transactions on Computational Social Systems, 10(3), 1394-1401. DOI: 10.1109/TCSS.2022.3189368