Concurrency and Computation: Practice and Experience

WILEY

SPECIAL ISSUE PAPER

A Secure and Communication-Efficient Federated Recommendation Method With Blockchain

Sheng Lu | Daming Huang | Zhehong Wang | Zheng Li | Hang Zhang | Wanchun Dou | Chen Tian

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Correspondence: Chen Tian (tianchen@nju.edu.cn)

Received: 15 May 2025 | **Revised:** 27 June 2025 | **Accepted:** 15 July 2025

Funding: This work was supported by the National Key Research and Development Program of China under Grant No. 2022YFB2702800 and the National Natural Science Foundation of China under Grant No. 92267104.

Keywords: blockchain | communication efficiency | fake model detection | federated learning | low-rank training | recommendation system | security

ABSTRACT

Due to the significant advantages of federated learning (FL) in privacy protection, federated recommendation systems (FedRSs) have garnered increasing attention by enhancing recommendation performance through local data training. However, most current FedRSs adopt a client-server communication architecture, which may lead to communication overload and single points of failure. Additionally, clients may face challenges due to limited communication resources and malicious attacks. To address the above challenges, we propose a Blockchain-assisted Federated learning method for Recommendation, called BFedRec, suitable for recommendation systems with high communication efficiency requirements. Specifically, BFedRec achieves the aggregation and distribution of recommendation models through a blockchain system, reducing reliance on central servers and alleviating communication bottlenecks and single points of failure. On this basis, BFedRec applies an innovative FL method that trains recommendation models directly on low-rank parameters to achieve efficient and secure data aggregation and distribution. Moreover, the flexibility of this aggregation and distribution strategy allows for scalable applications in other fields, such as blockchain-enabled software-defined network (SDN) management in on-chain and off-chain communication networks. Experimental results demonstrate that BFedRec outperforms existing methods on real datasets, significantly improving communication efficiency while effectively enhancing system security and robustness.

1 | Introduction

With the swift expansion of e-commerce and digital services, digitization has become deeply ingrained in people's daily lives. Nowadays, individuals spend a significant portion of their time online, exploring a diverse range of content tailored to their specific interests. Recommendation is essential for e-commerce [1] and digital service [2] providers by leveraging user data to offer personalized recommendations, thereby enhancing the overall user experience. It analyzes various types of data, including historical records and user interactions, to understand individual preferences and suggest relevant items or content [3, 4].

Centralized recommendation systems typically collect and consolidate user interaction data on a unified server. Nonetheless, this methodology poses a risk of disclosing private information that users may prefer to keep confidential. Unauthorized access to this data can lead to severe security threats and potential misuse by malicious entities, compromising user privacy. Accordingly, the centralized gathering of personal data has been

regulated through policies like the California Consumer Privacy Act (CCPA) [5].

Given these concerns, federated learning (FL) [6] is increasingly recognized as a viable remedy. A global model is collaboratively trained by clients under the framework of federated learning, while preserving the confidentiality of their private data. The coordination is overseen by a central server, which manages the interactions among the clients. During every training round, updates are sent by clients to the server, where they are merged to enhance the global model. This process repeats until the model converges. The development of federated recommendation systems (FedRSs) [7, 8] is a direct extension of the principles of FL. In typical cross-device scenarios, the training of FedRSs involves transferring models between multiple edge devices (e.g., smartphones, laptops) and a central server. FedRSs have gained increasing attention for improving recommendation performance while protecting user privacy by training models on local data.

However, conventional FedRSs assume ideal communication conditions between servers and clients, which is often not the reality. Additionally, FedRSs may face security issues, such as service vulnerabilities and malicious poisoning attacks from clients.

First, limited client-side communication resources pose a significant challenge for FedRSs. Clients, often located at the network edge, face bandwidth constraints and frequent data transmissions, leading to bottlenecks. Moreover, the increasing complexity and parameters of models in modern recommendation systems make their transmission more difficult [9, 10]. Additionally, clients participating in FedRSs typically have varying computational speeds and communication bandwidth capabilities due to differences in hardware and infrastructure [11]. These disparities can lead to stragglers, reducing the number of participants in training and thereby degrading system performance. Minimizing the size of transmitted information through compression is a common solution to reduce communication costs.

Despite various technologies aimed at enhancing communication efficiency, the outcomes are suboptimal. (1) Top-K compression minimizes the size of transmitted parameters by sparsification, yet clients incur extra expenses to organize and recognize updated data. (2) Singular Value Decomposition (SVD) compression transmits low-rank model updates via singular value decomposition, but these updates may lose their low-rank status after server-side aggregation, ultimately failing to decrease downlink communication costs. These compression methods involve encoding and decoding, which lead to considerable delays that may outweigh the per-bit communication time savings [12]. Additionally, they only optimize uplink communication costs without significantly improving downlink communication costs.

Second, security issues are also a common concern in FedRSs. The traditional FL framework presupposes that both servers and clients are reliable, which is often not the case in practical applications. In reality, malicious clients can submit corrupted updates, resulting in poisoning attacks. Recent research has demonstrated that FL is highly vulnerable to such attacks [13], with even a single Byzantine client capable of severely degrading the performance of the global model [14]. Furthermore, the centralized coordination and aggregation in FL confer a predominant role to the server, which makes the FedRS vulnerable to server failures or dishonest behavior.

In response to the challenges posed by malicious clients and server bottlenecks, significant research has been dedicated to improving the robustness and decentralization of FL. Various defense strategies have been proposed, including Multi-krum [13], Coordination-wise Median (CM) [15], Robust Federated Averaging (RFA) [16] and Bulyan [17], to safeguard against attacks from malicious clients.

Blockchain technology presents a viable solution for decentralized aggregation, tackling the issue of single points of failure in FL. Its features, such as tamper resistance, transparency, and traceability, make it attractive for securing FL systems. As a result, several blockchain-based FL frameworks have been developed. For example, some research suggests fully deploying blockchain on client devices [18–20]. However, these systems face high communication and storage overheads as a result of the substantial block sizes that contain both local model updates and global models in the continuously growing blockchain. The approach in Reference [21] mitigates this by enabling clients to remove older blocks in accordance with their local resource constraints, though this comes at the cost of reduced security and trust. Other studies explore deploying blockchain on edge nodes, which have more computational and storage capabilities compared to client devices [22]. Nevertheless, the system in ChainsFL [22] needs to distribute the data needed to validate the model, which compromises FL's privacy-preserving objectives, while the method in Reference [23] is susceptible to single points of failure (SPOF) due to centralized aggregation by the task publisher. Additionally, the utilization of widely adopted blockchain platforms such as Ethereum, in conjunction with smart contracts [24, 25], introduces high operational costs, as each blockchain interaction incurs gas fees. As a result, there is a need for a blockchain-based FL framework that is more cost-effective, reliable, and secure, capable of operating on resource-constrained client devices situated at the vulnerable network edge.

Based on the above discussion, we propose BFedRec, a blockchain-based federated edge learning approach specifically developed for recommendation systems that incorporates low-rank training. Inspired by BEFL [26], we integrate a blockchain network across edge nodes to reduce reliance on central servers. By utilizing a committee-based consensus mechanism with Verifiable Random Functions (VRF), model aggregation and distribution are performed in a decentralized manner at the edge nodes. To enhance BFedRec, we employ a low-rank FL method to improve communication efficiency and reliability. Specifically, we use low-rank structure training techniques to compress model parameters and reduce communication overhead. This technique trains a lightweight low-rank model while freezing most parameters, thereby reducing both uplink and downlink communication costs to improve communication efficiency. Additionally, to mitigate malicious poisoning attacks from clients, we introduce an anomaly detection mechanism. This mechanism captures the inherent correlations between model updates on the client side using mutual information (MI) and selects reliable updates based on their MI

values. The BFedRec method, with its decentralized, secure, and privacy-preserving features, demonstrates excellent versatility and scalability, making it applicable to various scenarios. Its core mechanism leverages blockchain technology and federated learning methods to reduce dependence on centralized servers while enhancing system robustness and efficiency through secure data aggregation and distribution. The primary contributions are delineated as follows:

- We present an innovative blockchain-based FRS that facilitates decentralized aggregation of recommendation models by utilizing a blockchain network deployed on edge nodes, thereby obviating the necessity for a central server in the processes of global model aggregation and distribution, improving system stability and security.
- We introduce a technique that focuses on optimizing low-rank structural parameters. This approach reduces communication costs in both uplink and downlink channels, contributing to efficient and scalable federated learning in recommendation systems.
- To enhance the reliability, we introduce a fake model detection mechanism that ensures only valid model updates are accepted by the blockchain network, thereby preventing malicious poisoning attacks from clients.

The subsequent sections of this paper are structured as follows: Section 3 offers an overview of the background and fundamental concepts relevant to this study. Section 4 presents the overview and workflow of the proposed BFedRec method. Section 5 describes the consensus mechanism and incentive mechanism. Section 6 introduces the low-rank training and fake model detection mechanisms. Section 7 describes the experimental configuration and the findings. Section 8 provides a summary of this paper.

2 | Related Work

This section reviews the related work on FedRSs and blockchain-based FL, with a focus on blockchain-assisted FedRSs.

2.1 | Federated Recommendation Systems

FedRSs have gained considerable attention as a major research focus in machine learning and recommendation systems. Early FL approaches, such as FCF [7] and FedRec [8], were developed for implicit and explicit feedback, respectively, using matrix factorization techniques. FedMF [27] integrates matrix factorization (MF) in the FL framework and employs homomorphic encryption (HE) to secure gradients uploaded to the server. MetaMF [28] introduces a distributed MF approach that leverages meta-learning to create models for predicting ratings and generating private item embeddings. HPFL [29] introduces a hierarchical personalized FL framework aimed at tackling heterogeneity, thereby enhancing recommendation performance. FedPerGNN [30] develops a method for maintaining a graph neural network (GNN) model for each user, capturing more intricate user-item relationships.

In the past few years, considerable progress has been achieved in FedRSs across various aspects. For instance, Ye et al. [31] and He et al. [32] proposed clustering-based FL frameworks to better group clients and tailor recommendations. Zhang et al. [33] addressed the cold start issue by decoupling item attributes from interaction data, allowing better recommendations for new items and users. Nguyen et al. [34] optimized both communication costs and security by incorporating low-rank training, while Qu et al. [35] introduced personalized privacy protection to enhance user control and security in federated systems. Additionally, Yan et al. [36] developed a privacy-protected recommendation framework utilizing a federated heterogeneous graph neural network, which improves both security and recommendation accuracy through a graph-based approach.

At the same time, communication efficiency is crucial in FL. Some studies focus on minimizing the size of the item latent matrix using meta-learning approaches [28]. LightFR [37] introduces a framework within a federated setting, aiming to cut down on communication costs while enabling faster online inference and lower memory usage. Khan et al. [38] designed an algorithm to tackle the challenges associated with payload in item data.

The transmission of updates to the server without adequate privacy protection measures exposes FL systems to security risks. Chai et al. [27] showed that in MF models using Federated Averaging (FedAvg) [6], adversaries could deduce users' rating information by accessing their gradients over two consecutive steps. One potential solution to address this challenge is the implementation of HE to protect parameters before they are transmitted to the central server [27, 39]. This approach ensures user rating privacy while maintaining the accuracy of recommendations. Nevertheless, it imposes considerable computational demands, involving encryption and decryption operations on the clients, along with aggregation on the central server. Liang et al. [40] propose enhancing FedRS performance through denoising clients, while Liu et al. [41] explore secure recommendation systems in cross-domain contexts. Recent research [42–44] highlights that FedRSs remain susceptible to poisoning attacks from malicious clients.

2.2 | Blockchain-Based Federated Learning

To tackle the problem of unreliable servers, blockchain-based FL has been identified as a promising approach. Blockchain's decentralized, peer-to-peer architecture facilitates the aggregation of global models, offering a solution to SPOF in conventional FL systems. Several studies [18, 19, 21, 22, 24] have shifted the aggregation process from central servers to blockchain nodes. Meanwhile, Qu et al. [45] have removed the aggregator while enabling clients to perform the aggregation of model updates themselves.

Most blockchain-based FL methods integrate model verification to make the global model robust. The verification techniques vary based on the assumptions about blockchain nodes. In certain frameworks, it is assumed that clients not only participate in the FL process but also join the blockchain network, using their training data for validation purposes [19, 21, 46]. This makes the verification accuracy of the local model helpful for the aggregation stage. Li et al. [21] proposed a method involving K-fold

cross-validation, where a group of K participants assesses model updates using their own datasets to determine which updates should be included in the global aggregation process. Similarly, Lu et al. [46] developed a technique to select updates that meet certain accuracy criteria, which are then used to refine the global model. In blockchain-based FL systems, even when nodes cannot access training data, accuracy-driven validation continues to be applied. For instance, Mugunthan et al. [25] used smart contracts and a cross-validation process, where clients test each other's model updates on their local datasets and return accuracy values to calculate contribution scores, which are then used for weighted aggregation.

Although client-based model evaluation can reduce the risk of malicious updates, it significantly increases communication overhead and is dependent on the clients' active participation. As a result, recent blockchain-based FL frameworks have incorporated the latest developments in robust FL to mitigate the impact of malicious clients. SPDL [20], Biscotti [18], and Omnilytics [24] utilize the Multi-krum algorithm as a verification mechanism, allowing model updates close in Euclidean distance to pass. Biscotti introduces noisers, verifiers, and aggregators, where noisers add Differential Privacy (DP) Gaussian noise to updates, verifiers apply Multi-krum for validation, and aggregators securely aggregate the verified updates. In a similar approach, SPDL incorporates DP Gaussian noise into the local updates and utilizes Multi-krum during the aggregation phase, offering theoretical convergence guarantees for private and secure FL. Omnilytics incentivizes honest clients and penalizes malicious ones through smart contracts.

While well-established platforms like Ethereum support the secure execution of FL tasks via smart contracts, they come with high gas fees for each blockchain interaction. Edge nodes offer significant promise due to their computational, communicative, and storage capabilities. However, blockchain-based FL solutions targeting edge environments still struggle with addressing security concerns from malicious clients, unreliable servers, and the high communication demands placed on clients, which are resource-constrained.

2.3 | Blockchain-Assisted Federated Recommendation Systems

Blockchain-based federated recommendation systems have gained attention for their ability to address privacy and security concerns in personalized recommendation tasks. By integrating blockchain with FL, these systems allow collaborative model training while ensuring that sensitive data remains decentralized and secure. Blockchain technology plays a key role by providing transparency and integrity in the recommendation process, where model updates are securely recorded and tampering is prevented. This decentralized approach ensures that participants, including users and service providers, can trust the system without the need to share sensitive data directly, making it particularly suitable for applications in privacy-sensitive domains such as healthcare, financial services, and consumer electronics [47].

Several recent works have proposed blockchain-based federated recommendation systems, each focusing on improving communication efficiency, privacy, and security. Ali et al. [48] proposed a blockchain-empowered asynchronous FL method which leverages blockchain to ensure tamper-proof aggregation and decentralizes the model training process. While this approach reduces risks associated with single-point failures, it still faces communication bottlenecks and scalability issues due to frequent model synchronization. Similarly, FLRM [49] incorporates blockchain to guarantee data security and transparency in financial institutions, but struggles with high communication overhead, especially in large-scale systems. Moreover, frameworks such as Secure and Privacy-Preserving Decentralized FL combine blockchain with homomorphic encryption to protect data privacy during model training [50]. Although these methods enhance privacy, the computational complexity introduced by encryption leads to higher latency, which can hinder performance in real-time applications.

In contrast to these existing methods, BFedRec introduces significant improvements in communication efficiency, scalability, and security. By utilizing low-rank training and decentralized aggregation, BFedRec drastically reduces communication overhead, which makes it more efficient and scalable compared to other blockchain-based federated recommendation systems. While other approaches rely heavily on blockchain for model updates and synchronization, BFedRec minimizes the frequency of communication, resulting in lower latency and better overall performance. Additionally, BFedRec enhances security by decentralizing the aggregation process, effectively mitigating risks such as model poisoning and tampering without the need for computationally expensive encryption techniques. This makes BFedRec particularly well-suited for large-scale, real-time recommendation systems, where both efficiency and security are critical.

3 | Preliminaries

The foundational concepts relevant to the proposed method are introduced in this section, including blockchain and FL for recommendation systems.

3.1 | Blockchain

Blockchain represents a distinctive type of distributed ledger system that ensures the integrity and permanence of data or transactions. It functions on a decentralized network, integrating core principles like transparency, cryptography, consensus mechanisms, immutability, and enhanced security. Blockchain leverages cryptography to secure data, verify transactions, and protect stored information using hash functions, digital signatures, and encryption. All transactions on the blockchain can be accessed by all participants and are transparent, and once added, they are almost impossible to modify, ensuring data integrity and traceability. Consensus mechanisms enable distributed nodes to reach an agreement on the validity and sequence of transactions. Smart contracts are self-executing protocols encoded into the blockchain, which can automatically execute transactions and enforce terms without intermediaries. Blockchain facilitates trust by eliminating centralized institutions, and its decentralized and cryptographically secure design makes it resistant to tampering by malicious actors.

Our method employed the InterPlanetary File System (IPFS) to store data and the Verifiable Random Function (VRF) to verify committee selection. The foundational concepts in our proposed method are as follows:

- IPFS: IPFS is a peer-to-peer distributed file system that can provide permanent decentralized data storage. Data is stored in IPFS nodes and retrieved using a unique hash index derived from the data, further secured by the tamper-resistant features of blockchain technology.
- VRF: VRF is a public key pseudo-random function that can prove its random output given the input. VRF provides a non-interactive method for blockchain nodes, enabling them to independently determine whether they are selected as committee members.

3.2 | Federated Learning for Recommendation Systems

In an item-based FedRS, N items and M users are set commonly, where each user u has an interaction set $O_u = \{(i, r_{ui})\}$. r_{ui} represents a rating or interaction value. Users aim to collaboratively build a recommendation system while preserving privacy. This scenario is well-suited for horizontal FL, where each user actively participates.

The system aims to generate a top K list, listing items related to user preferences that have not yet been interacted with. This problem can be mathematically defined as finding a global model represented by parameters θ to minimize the global loss function $\mathcal{L}(\cdot)$.

$$\mathcal{L}(\theta) := \sum_{u=1}^{M} w_u \mathcal{L}_u(\theta) \tag{1}$$

Here, w_u is the relative weight, θ is the global parameter, and $\mathcal{L}_u(\cdot)$ is the loss function on u's device. To achieve our objective, we utilize a federated matrix factorization (FedMF) [7] model as the core model, with the local loss function for user u defined as follows:

$$\mathcal{L}_u(p_u, Q) := \sum_{(i,r_{ui})\in O_u} \ell\left(r_{ui}, (Q^T p_u)_i\right) + \frac{\lambda}{2}\|p_u\|_2^2 + \frac{\lambda}{2}\|Q\|_2^2 \tag{2}$$

In this model, the local parameters of user u consist of the user embedding vector p_u and the item embedding matrix Q. The loss function $\ell(\cdot)$ quantifies the difference between actual and predicted values. Typically, we use MSE as the loss function, but other loss functions like cross-entropy loss can also be used. The parameter λ is the regularization term to prevent overfitting.

FedAvg [6] is a typical FL algorithm where local model updates are averaged, enabling privacy-preserving training without centralized data. At each round, the server distributes the current item embedding matrix $Q^{(t)}$ to a group of users $S(t)$, and then each user performs local SGD updates on their datasets. The local update rule for user u is as follows:

- User embedding vector $\mathbf{p}_u$:

$$\mathbf{p}_u^{(t+1)} = \mathbf{p}_u^{(t)}(1 - \eta\lambda) + \eta Q^{(t)\top}\left(\mathbf{r} - \hat{\mathbf{r}}^{(t)}\right) \tag{3}$$

TABLE 1 | Notations.

Symbol	Description
M	Number of users
N	Number of items
O_u	Interaction set of user u
r_{ui}	Rating of user u on item i
θ	Global model parameter
w_u	Weight of user u
$\mathcal{L}(\cdot)$	Global loss function
$\mathcal{L}_u(\cdot)$	Local loss function of user u
p_u	User embedding vector of user u
Q	Item embedding matrix
$\ell(\cdot)$	Loss function
λ	Regularization term
$S(t)$	Group of users at round t
η	Learning rate
$\Delta Q_u^{(t)}$	Local item embedding matrix update of user u at round t
$A_u^{(t)}$	low-rank approximation of $\Delta Q_u^{(t)}$
$B_u^{(t)}$	projection matrix of $Q^{(t)}$
K	Committee size
(SK, PK)	VRF key pair
(sk, pk)	Signature commitment key pair
s_i	Stake of node i
S	Total stake
G_{IPFS}^t	IPFS address of the global model at round t

- Item embedding matrix Q:

$$Q^{(t+1)} = Q^{(t)} - \eta\left(\lambda Q^{(t)} - \left(\mathbf{m}\left(\mathbf{r}_u - \hat{\mathbf{r}}_u\right)\right)\mathbf{p}_u^{(t)\top}\right) \tag{4}$$

Next, the user will upload the local item embedding matrix update $\Delta Q_u^{(t)} = Q^{(t+1)} - Q^{(t)}$, and the server will aggregate these received updates and update the global model:

$$\theta^{(t+1)} = \theta^{(t)} + \frac{\sum_{u\in S(t)} w_u \Delta Q_u^{(t)}}{\sum_{u\in S(t)} w_u} \tag{5}$$

This process is repeated until the algorithm converges. The main notations are summarized in Table 1.

4 | A Secure and Communication-Efficient Federated Recommendation Method With Blockchain

In this section, the framework of our proposal is introduced, and the specific tasks of different participants are explained, including the initialization work of the central server, the local low-rank training method for the user client, and the secure aggregation and consensus of models using the blockchain network.

4.1 | Overview

The framework of BFedRec is represented in Figure 1. BFedRec involves several key participants: User clients, a central server, and a group of edge nodes deploying blockchain technology. The goal is to achieve communication-efficient and secure federated recommendation. The roles of these participants are described as follows:

- *Central Server*: In traditional cross-device FedRSs, the central server initializes the global model and coordinates client training. However, this centralized approach can lead to high communication loads on the server and introduce single points of failure, reducing system reliability. To address these issues, BFedRec uses blockchain technology to offload the server's workload and enhance system security. Consequently, the central server initializes the recommendation model at the start of the FL task and publishes it to the blockchain network.
- *Blockchain Nodes*: To mitigate the high load and vulnerability associated with centralized aggregation of the global model, we consider distributing the server's workload across multiple entities to achieve a more secure decentralized model aggregation. We choose edge nodes for this purpose due to their robust computational, storage, and communication capabilities. These edge nodes, also referred to as blockchain nodes, run and maintain the blockchain system, aggregating and verifying recommendation models. Each edge node supports a global model and aggregates model updates from the client while detecting false model gradients. The security and reliability of this process are protected by the blockchain network.
- *Clients*: Users interact with the blockchain through their clients, which collect local user data to train recommendation models. Before iterative training, clients retrieve the original model from the blockchain network to initialize their local models. During each round of FL, clients download the latest global update from the blockchain network, use local data for training. Then the trained updates are uploaded to the blockchain. The training process employs low-rank structures to decompose the recommendation model update, reducing communication load.

To choose blockchain nodes, we think that a trusted entity manages a permissioned blockchain network through certified edge devices like base stations. Once nodes are registered, they are assigned a VRF key pair (SK, PK) for committee selection, a key pair for signature commitment (sk, pk), and an initial stake.

The blockchain structure among edge nodes consists of a sequence of blocks linked by hash pointers. Each block comprises a header containing the index, the hash of the preceding block, and a timestamp, as well as a body containing transactions. The first block, known as the genesis block, contains essential system information. The FL tasks shared by the task publisher include the IPFS link to the initial model G^0_{IPFS} and relevant FL hyperparameters. Each update to the model, signed by a client, becomes a transaction. The model updates intended for global aggregation,

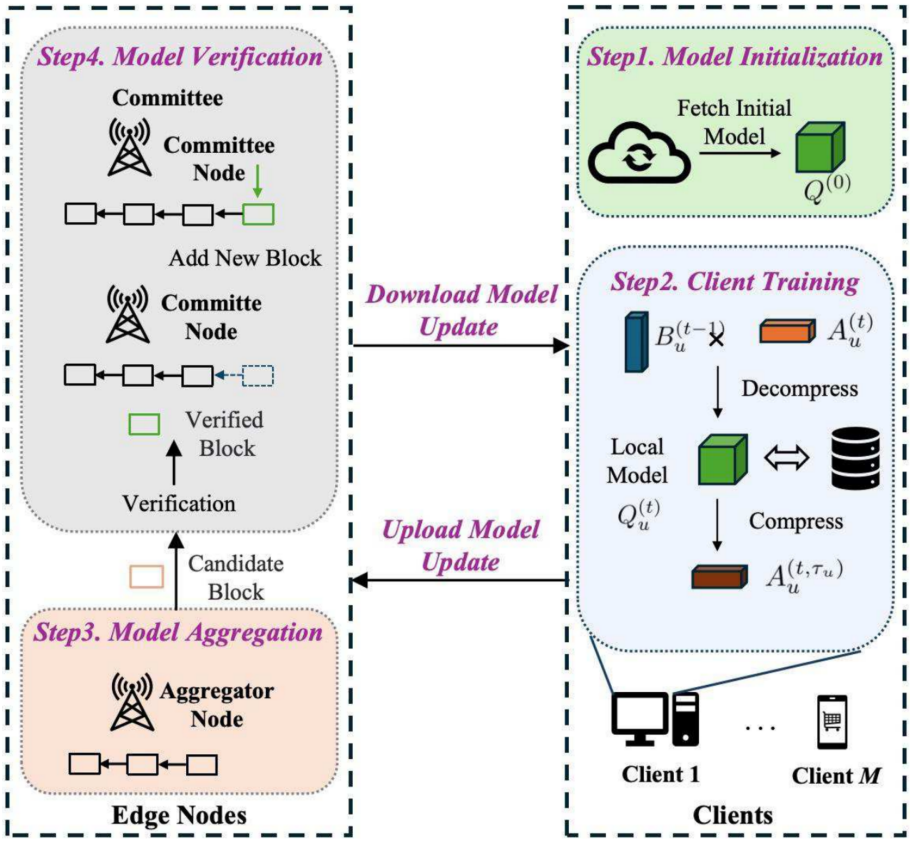

FIGURE 1 | The framework of BFedRec.

along with the IPFS link to the aggregated model, are included in the body of a candidate block. Upon confirmation, the local model updates contained in the candidate block are discarded to optimize storage, as they will not be useful for future rounds. Instead, the signatures of committee members who approved the block, the block header, and the model IPFS address are recorded.

4.2 | Workflow

The workflow of BFedRec is represented in Algorithm 1. The central server initializes the FL task by publishing the initial model to the blockchain. Clients retrieve the initial model IPFS address from the nearest active blockchain node and download the initial model from IPFS. During iterative training rounds, clients download the latest update from the blockchain network and train local models using private data. Then the compressed local update and signature are uploaded as a transaction to the nearest blockchain node, where a new verified block will be formed and added by aggregating and verifying the uploaded updates.

- *Step 1: Model Initialization*. The FL tasks and the initial recommendation model, stored on IPFS, are first made available on the blockchain by the central server. Clients retrieve the IPFS address of the initial model by querying the nearest active blockchain node and downloading the model using this address. This blockchain node also serves as the corresponding node for clients to upload and download models later. This approach alleviates the high communication overhead and potential single points of failure associated with centralized servers in FL, shifting model aggregation to decentralized blockchain nodes. After this initial phase, the central server becomes redundant and is no longer required for subsequent FL processes.
- *Step 2: Client Training*. In each training iteration, clients obtain the IPFS link to the most recent global model by querying the closest blockchain node. Clients use their private data for local model training. A low-rank structure is adopted to decompose the item embedding matrix for model compression. After local training, clients send the compressed model update and signature as a transaction to the nearest blockchain node.
- *Step 3: Model Aggregation*. During the model update collection phase, each blockchain node gathers model update transactions from clients. These nodes verify the authenticity of the signatures and model updates before propagating the transactions across the blockchain network and adding them to the blockchain. The latest block typically includes multiple model updates and client signatures. When a node collects a sufficient number of transactions, it competes to calculate the new global model update, records its IPFS address, and forms a candidate block. These candidate blocks are subsequently transmitted to the committee. The committee validates the global model through voting and propagates the verified block to the blockchain network.
- *Step 4: Model Verification*. In the model verification phase, committee nodes validate the update correctness from the global model via a voting process. If a candidate block secures over two-thirds of the committee's approval votes, it is considered validated and is transmitted to the whole network. The validated block replaces the original local model updates and includes the signatures of the committee nodes. Upon receiving the validated block, all other blockchain nodes verify the committee signatures and add the block to their local chains. Any other candidate blocks for that round are invalidated and removed. If consensus is not reached within a specified time, the committee discards the candidate block and awaits the next one. If a candidate block remains unconfirmed after multiple voting rounds, the process will be restarted by a new committee.

ALGORITHM 1 | BFedRec Workflow.

Server publishes IPFS address of the initial model G^0_{IPFS} to the blockchain;
Each client retrieves G^0_{IPFS} from the nearest blockchain node;
for $t = 0, 1, 2, \ldots, T$ **do**
  for ***each*** *client* ***in parallel*** **do**
    Retrieves the latest global model from the blockchain;
    Updates the model locally using private data according to Algorithm 3;
    Sends the update to the nearest blockchain node;
  Blockchain nodes aggregate model updates and generate a candidate block;
  Blockchain nodes broadcast the candidate block to the committee;
  Committee verifies the candidate block according to Algorithm 2;
  Nodes add the verified block to their local chain.

5 | Stake-Based Consensus Mechanism

In this section, the stake-based consensus mechanism is introduced to elect committee members for model aggregation. We discuss the committee election protocol and the hyperparameters α and K used in the consensus mechanism. We also introduce incentives for committee members and the reward distribution mechanism.

5.1 | Committee Election

In our stake-based consensus mechanism, the blockchain network uses VRF to set up a committee constructed by random subsets in a non-interactive manner. Each node generates a unique random hash using its secret key SK_i and a public seed, normalizes it to obtain r, and checks if it is the candidate committee member. The election protocol selects nodes based on their stake proportion. If the generated r satisfies $\frac{r}{s_i} < \frac{\alpha K}{S}$, Node i is selected as a potential committee member and transmits qualifying data (hash and proof) to the network, where α is a hyperparameter, K is the committee size, s_i is the node stake, and S is the total network stake. The value r produced by the VRF is a distinct

random number uniformly distributed within the interval [0, 1]. The probability can be calculated as follows:

$$p_i = \Pr\left(\frac{r}{s_i} < \frac{\alpha K}{S}\right) = \Pr\left(r < \frac{\alpha K s_i}{S}\right) = \frac{\alpha K s_i}{S} \tag{6}$$

As shown above, nodes with a larger stake have a greater chance of being chosen for the committee. This effectively mitigates Sybil attacks, where a malicious entity presents multiple identities to control a large portion of the system. When the committee reaches the correct size, the committee formation phase ends, and the first K nodes become committee members. Any node can obtain committee details and authenticate the committee members' identities by using the VRF public key, the most recent blockchain block, and their respective stakes.

The hyperparameter α controls the expected number of nodes that may be selected as candidate committee members. Since each node follows a Bernoulli distribution to become a candidate committee member, we denote the trial for node i as X_i, where $X_i = 1$ with probability p_i indicates a successful trial, and $X_i = 0$ with probability $1 - p_i$ indicates a failed trial. The sum of successful trials $X = \sum_{i=1}^{n} X_i$ has an expected value calculated as follows:

$$E(X) = \sum_{i=1}^{n} p_i = \alpha K \tag{7}$$

By leveraging the Chernoff bound for a series of independent Bernoulli trials, we derive

$$\Pr(X \leq (1-\delta)\alpha K) \leq e^{-\alpha K \delta^2/2} \tag{8}$$

for any $0 \leq \delta \leq 1$. Setting $\delta = \frac{\alpha - 1}{\alpha}$, where α is in the range $(1, \frac{n}{K}]$, we get

$$\Pr(X \leq K) \leq e^{-K(\alpha-1)^2/2\alpha} \tag{9}$$

To ensure a sufficient number of candidate committee members, the likelihood of having fewer than $K + 1$ candidate members should be minimized.

5.2 | Consensus Rules

The consensus protocol is a key element in ensuring the security of blockchain technology. In the model verification phase, we use the following five consensus rules:

- I_1: When new candidate blocks are submitted, the members of the committee first verify that sufficient pending transactions have been gathered and ensure that the signature on the IPFS address originates from the aggregator.
- I_2: Once these conditions are met, the committee members use the secure aggregation protocol to calculate the global model, which they then compare against the IPFS value. If the results align with the aggregator's output, they approve the block, signing both the header and the IPFS address.
- I_3: If more than two-thirds of the committee members agree, the candidate block is considered valid, and the transactions within it are replaced by the signatures of the supporting members.
- I_4: Voting on each candidate block must be completed within a predefined time limit. If the block fails to receive enough votes within the allotted time, it is discarded, and the committee proceeds to the next candidate block.
- I_5: Each committee has a set maximum number of voting rounds. If no candidate block is confirmed within these rounds, a fresh committee will be formed to evaluate the next batch of candidate blocks.

Rule I_3 relies on a two-thirds consensus threshold, which is rooted in the principle of Byzantine Fault Tolerance (BFT). In decentralized systems such as blockchain, certain nodes may experience failures or engage in malicious actions, causing inconsistencies. BFT algorithms can tolerate up to one-third of node failures, meaning a two-thirds threshold ensures that the system can still reach consensus even if a third of committee members are unresponsive or compromised.

Once a block is approved, the rewards associated with the stake are distributed among the committee members and the entity responsible for generating the block, ensuring that BFedRec does not include outdated updates in the model. The model verification method is summarized in Algorithm 2.

ALGORITHM 2 | Model Verification.

Input: The last block *seed*, stake of each node s_i, total stake S, committee size K, hyperparameter α, the candidate block $cBlock$, the maximum voting time $MaxVoteDuration$, the maximum vote round $MaxStep$.
Output: The verified block $vBlock$;

Committee Constitution:
for *each* *node i* **do**
  hash, proof = VRF(SK_i, public seed);
  if $\frac{hash}{s_i} < \frac{\alpha K}{S}$ **then**
    node i becomes a candidate committee member;
    Send the (hash, proof) to the network;
Committee Voting:
while *True* **do**
  Initialize the current voting round $step = 1$;
  while $step < MaxStep$ **do**
    if *voting time* $< MaxVoteDuration$ **then**
      Committee nodes vote for $cBlock$;
      if *agreements* $> 2/3$ **then**
        The $cBlock$ is verified successfully, and form $vBlock$;
        Distribute rewards to committee members and block generator;
        return;
      else
        The $cBlock$ is verified unsuccessfully, and it is discarded;
        $step$++ and wait for a new candidate block;
  Committee reconstruction;

5.3 | Incentive Mechanism

To further enhance the system's security and fairness, we employ a reward and penalty-based incentive mechanism designed to ensure all participants adhere strictly to the consensus protocol while effectively mitigating potential malicious behavior. In our approach, the reward mechanism not only incentivizes block generators but also covers the verification and voting processes of committee members. Committee members receive rewards proportional to their stakes in the committee, encouraging active participation. This economic incentive not only increases node participation but also reduces the motivation for malicious behavior. To further enhance security and prevent potential attacks, we introduce a strict penalty mechanism. If a node is detected violating the protocol during the consensus process (e.g., engaging in malicious behavior or refusing to vote), its staked assets will be partially or fully forfeited. The forfeited assets can be used to reward other honest nodes, creating a positive incentive to follow the rules.

Since both the reward and penalty mechanisms are tied to the nodes' stakes, the system exhibits natural resistance to Sybil attacks. Even if an attacker attempts to participate in the consensus by forging multiple fake nodes, their total stake proportion remains unchanged, thus not significantly increasing the probability of being selected as a committee member.

6 | Low-Rank Federated Learning

In this section, a low-rank federated learning strategy is introduced to reduce communication costs and enhance system efficiency. Then, we present the secure aggregation protocol used in the blockchain network to maintain the security and integrity of the FL process.

6.1 | Communication-Efficient Training

In FedRSs, communication costs are largely driven by the updates to the item embedding matrix. Given the limited computational and communication resources of user clients, an efficient compression algorithm is critical for reducing data uploads and minimizing communication overhead.

As shown in Equation (4), the update of the item embedding matrix consists of a regularization term and a rank-1 matrix. Inspired by the work in Reference [34], we take advantage of the small regularization parameter λ by restricting the updates to a low-rank structure, thereby reducing communication costs. Specifically, the local model update $\Delta Q_u^{(t)}$ is parameterized as:

$$\Delta Q_u^{(t)} = B_u^{(t)} A_u^{(t)} \tag{10}$$

where $B_u^{(t)} \in \mathbb{R}^{d\times r}$ and $A_u^{(t)} \in \mathbb{R}^{r\times N}$. Since $r < N, d$, this parameterization reduces the communication load by a factor of $\frac{N\times d}{N\times r + r\times d}$. This parameterization can be understood as projecting the high-dimensional parameter matrix $\Delta Q_u^{(t)}$ into a low-dimensional subspace. $B_u^{(t)}$ represents the projection matrix, and $A_u^{(t)}$ represents the low-rank parameter matrix after projection.

To further minimize downlink communication costs, only $A^{(t)}$ is transmitted between the client and the blockchain system. The blockchain aggregates the updates while maintaining low-rank downlink transmission. At the start of each local training round, the client updates its $B_u^{(t)}$, which remains fixed during the training process. The aggregation result can be captured as follows:

$$\Delta Q^{(t)} = B^{(t)}\left(\sum_{u\in S} A_u^{(t)}\right) \tag{11}$$

In the initial round of training, clients download the $Q^{(0)}$ to initialize their local models. After receiving this global model, clients begin the local training process. In each subsequent round, the server aggregates the local updates $\{A_u^{(t-1,\tau_u)}\}$ submitted by all clients to update $A^{(t)}$, which is then transmitted back to the clients. Upon receiving $A^{(t)}$, clients use it to update their local models $Q_u^{(t)}$. This update involves combining the global update $A^{(t)}$ with the local model $Q_u^{(t)}$, followed by extracting the updated $B_u^{(t)}$, which remains fixed in the subsequent local training rounds.

At the beginning of each local training round, clients reset $A_u^{(t,0)} = 0$ and proceed with local training to optimize the model parameters $\theta_u^{(t,0)} = \{A_u^{(t,0)}, p_u^{(t,0)}\}$. During this process, clients perform multiple updates based on their local data and model parameters, continuously optimizing the local model. The local model parameters $\{A_u^{(t,\tau_u)}\}$ are uploaded to the server once training is completed. After collecting all the local updates, the server aggregates them, triggering a new round of global model updates.

The process allows the system to continuously optimize both global and local models while keeping communication costs low. Algorithm 3 outlines the low-rank training process in detail.

ALGORITHM 3 | Low-Rank Federated Learning.

Input: Initial item embedding matrix $Q^{(0)}$ or latest model update $A^{(t)}$; current round t; rank r; learning rate of client η;
Output: The local model updates $\{A_1^{(t,\tau_u)}, \dots, A_M^{(t,\tau_u)}\}$;

Sample a client subset $S^{(t)}$;
for ***each*** *client* $u \in S^{(t)}$ ***in parallel*** **do**
  if $t = 0$ **then**
    Download $Q^{(0)}$;
    Initialize $Q_u^{(t)} = Q^{(0)}$;
  else
    Download $A^{(t)}$;
    Merge $Q_u^{(t)} = Q_u^{(t-1)} + B_u^{(t-1)} A^{(t)}$;
  Update $B_u^{(t)}$ from $Q_u^{(t)}$;
  Initialize $Q_u^{(t,0)} = Q_u^{(t)}$ and $A_u^{(t,0)} = 0$;
  Set trainable parameters $\theta_u^{(t,0)} = \{A_u^{(t,0)}, p_u^{(t,0)}\}$;
  for $k = 0, \dots, \tau_u - 1$ **do**
    Perform local update $\theta_u^{(t,k+1)} = \mathrm{GSD}\left(\theta_u^{(t,k)}, \nabla\mathcal{L}_u\left(\theta_u^{(t,k)}\right), \eta\right)$;
  $p_u^{(t+1)} = p_u^{(t,\tau_u)}$;
  Upload $A_u^{(t,\tau_u)}$ to the nearest blockchain node;

6.2 | Secure Aggregation

The way models are trained in FL can make them vulnerable to attacks from malicious clients who send corrupted model updates to disrupt the training process. While low-rank approximation techniques help reduce communication costs, they introduce biases in the model updates, further complicating the identification of malicious contributions. This makes maintaining security without sacrificing performance challenging.

As noted by prior research [51], mutual information (MI) between honest clients increases over time, unlike Euclidean distance measures. Leveraging this, a robust aggregation method is proposed to distinguish between honest and malicious clients. MI quantifies the shared information between two random variables and serves as a powerful tool for detecting dependencies in model outputs that might not be apparent using traditional metrics.

In our model, each client's local stochastic gradient descent (SGD) output is treated as a random vector. Assuming the outputs F_i and F_j from clients i and j follow Gaussian distributions with variances σ_i^2 and σ_j^2, the MI between them, $MI_{i,j}$, is calculated as:

$$MI_{i,j} = -\frac{1}{2}\log(1-\rho_{ij}^2) \tag{12}$$

where ρ_{ij} is the Pearson correlation coefficient between F_i and F_j, computed as:

$$\rho_{ij} = \frac{E[(F_i - E[F_i])]E[(F_j - E[F_j])]}{\sigma_i^2\sigma_j^2} \tag{13}$$

A high ρ_{ij} value implies a strong correlation, indicating honest contributions, while a low correlation suggests potential malicious updates.

To quantify the similarity between updates, we compute MI scores for all client pairs and average them. Model updates that deviate significantly—either too low (indicating malicious intent) or too high (leading to redundancy)—are flagged. We utilize the median absolute deviation (MAD) instead of the standard deviation (SD) to more reliably measure the distance of each MI score from the central value. The MAD is normalized as:

$$MI_{\text{madn}} = \frac{MI_{\text{mad}}}{0.6745} \tag{14}$$

where the MAD value for the normal distribution is 0.6745. Model updates within two standard deviations of the median are selected for aggregation, ensuring robustness against outliers. The global model is then transmitted using Nesterov's momentum.

After the aggregation process, the momentum m_{t+1} and global model W_{t+1}^g are stored on the IPFS. Then, the candidate block that includes the IPFS address is generated by the aggregator and submitted to the committee.

7 | Experiments and Results

In this section, extensive experiments are conducted to evaluate the performance of our proposed method. The experiments seek to answer the following research questions:

- *RQ1*: How does our proposed method perform in FL, with a focus on recommendation accuracy and the efficiency of the training process in communication-constrained environments?
- *RQ2*: What is the performance of our method in terms of blockchain characteristics, specifically its latency in model aggregation and scalability under varying network conditions?
- *RQ3*: How does our method ensure robust security against various attacks, including poisoning and Byzantine attacks, while maintaining system integrity in FL?

7.1 | Experimental Setup

7.1.1 | Datasets

We evaluated our method on four publicly available datasets: MovieLens-1M [52], and LastFM-2K [53]. Table 2 summarizes the statistics of these datasets. Following common practices in recommendation systems, we first retained users with a minimum of 20 interactions and then transformed the explicit ratings into implicit feedback signals.

In FL settings, different users always interact with different items, leading to data quantity heterogeneity. To measure this heterogeneity, we show the maximum, minimum, average, standard deviation, and variance of local observations for each client in Table 3. The difference between the minimum and maximum number of local observations indicates a highly uneven distribution of data among clients. Specifically, the LastFM-2K and MovieLens-1M datasets exhibit significant data heterogeneity. The large standard deviation and variance in both datasets suggest substantial differences in data quantity among clients. Some clients have few interactions, while others have a large amount of interaction data. This disparity means that during FL, some clients may not provide enough data to support effective training, while clients with larger data volumes may dominate the training process. Due to this high heterogeneity in data distribution, FL may face challenges such as reduced training efficiency, uneven

TABLE 2 | Statics of the datasets.

Dataset	#Users	#Items	#Interactions	#Sparsity (%)
MovieLens-1M [52]	6,040	3,706	1,000,209	95.53
LastFM-2K [53]	1,600	12,454	185,650	99.07

TABLE 3 | Data quantity heterogeneity.

Dataset	#Min	#Max	#Avg.	#St. dev.	Var.
MovieLens-1M [52]	20	2,314	165.6	192.73	37,145
LastFM-2K [53]	5	2,609	116.03	240.07	57,635

model performance, and varying local device training times compared to traditional centralized learning. Consequently, data heterogeneity can adversely affect the training efficiency and final model performance of FedRSs, especially in the presence of data imbalance.

7.1.2 | Baselines

For the base model, we adopted NCF [54], an advanced deep neural network-based collaborative filtering model for modeling the features of items and users. We then compared several FL methods that primarily address general settings in inconsistent scenarios. In our experiments, these methods were applied to NCF-based recommendation tasks. We compared BFedRec with two baselines (i.e., centralized baseline and independent baseline) and six state-of-the-art baselines, namely FedSGD, FedAvg [6], FedProx [11], MOON [55], SCAFFOLD [56], and FedDyn [57]. The independent setting means training each client separately using local datasets. The centralized setting means sacrificing data privacy and training the global model using the entire dataset.

- *FedSGD* [6] is a stochastic gradient descent-based FL approach, where clients update their models with a single gradient descent step each round.
- *FedAvg* [6] averages local model updates and enables clients to perform multiple training steps before sending updates to the server for improved convergence and efficiency.
- *FedProx* [11] introduces a proximal term in the training process on each client, which constrains the local optimization process to prevent excessive drift from the global model.
- *MOON* [55] is a contrastive learning based FL method, enhancing the generalization ability of the model through contrastive loss.
- *SCAFFOLD* [56] accelerates the convergence of federated learning by correcting the bias in local updates.
- *FedDyn* [57] improve FL performance by dynamically adjusting the influence weights of the local model and the global model.

We also compared BFedRec with several federated recommendation methods that are more suitable for recommendation tasks:

FedNCF [58]: This approach adapts Neural Collaborative Filtering (NCF) to the federated setting. Each client updates its user embedding locally, while item embeddings and the scoring function are shared with the server for global aggregation.

Federated Reconstruction (FedRecon) [59]: A recent personalized federated learning framework evaluated here in the context of matrix factorization. Unlike traditional methods, FedRecon reinitializes user embeddings in each round, retraining them from scratch rather than inheriting from previous rounds.

Meta Matrix Factorization (MetaMF) [28]: A distributed matrix factorization method that leverages a meta-network to produce both the rating prediction module and private item embeddings, enabling more flexible and adaptive recommendation modeling.

In addition to verifying different FL methods, two compression methods were also selected for comparison, including BFedRec-TopK and BFedRec-SVD, which utilize Top-K and Singular Value Decomposition (SVD) compression, respectively. BFedRec-TopK utilizes sparsification to minimize transmission size by representing updates as sparse matrices. BFedRec-SVD leverages SVD to provide compressed updates with low-rank characteristics. For the blockchain, the Biscotti [18] is selected as the benchmark for the same goal of leveraging blockchain technology to defend against malicious clients.

7.1.3 | Evaluation Metrics

For the four experimental datasets, we apply the standard leave-one-out evaluation method to construct the test set [1]. Specifically, for each user, we use all of their interactions for training, while reserving their most recent interaction for testing. During the testing phase, we randomly select 99 items that the user has not interacted with and rank the test item among these sampled items.

We utilize users' entire interaction history for training, leaving only the most recent one for testing. In the testing phase, 99 items that the user hasn't engaged with are randomly selected, and the test item is ranked among these choices.

To assess the effectiveness and performance of our model, two metrics are employed, including Normalized Discounted Cumulative Gain (NDCG) and Hit Rate (HR). These metrics are typically limited to a specific rank (such as the top k items) to focus on the relevance of the most highly ranked results. HR checks whether the test item appears in the top k ranking, while NDCG evaluates ranking quality by taking both the position of the test item and the accuracy of the ranking into account. These metrics help determine whether the model ranks preferred items higher, providing an indication of its ranking effectiveness. In our experiments, we set k to 10 for the recommendation list.

7.1.4 | Implementation Details

In the experiments, the embedding sizes for both users and items are set to 64. A basic SGD optimizer is employed to train local models. M clients are randomly chosen without replacement during each round, both within and across rounds.

7.2 | Performance Analysis in FL for Recommendation Accuracy and Training Efficiency (RQ1)

7.2.1 | Recommendation Performance

To evaluate the recommendation performance in communication-constrained conditions, we first compared BFedRec with eight baseline methods and BFedRec variants using Top-K compression and SVD under the same communication budget. For the MovieLens-1M and LastFM-2K datasets, we set the user and item embedding dimensions of the base model to 8, while fixing BFedRec's embedding size at 64 and rank at 4. This setup ensures that all methods have roughly equal transmission sizes across the two datasets, enabling a fair comparison. Similarly, for LastFM-2K, we used the same embedding dimensions and rank settings.

Table 4 shows the HR and NDCG metrics under the same transmission scale. With equal communication budgets: (1) The centralized setting achieves the highest accuracy, while the independent setting has the lowest, representing the upper and lower bounds of accuracy, respectively. (2) BFedRec consistently outperforms all FL baseline methods and federated recommendation methods across both datasets. Specifically, BFedRec achieves the highest HR and NDCG scores among all FL methods, with HR = 0.61 and NDCG = 0.38 on MovieLens-1M, and HR = 0.74 and NDCG = 0.55 on LastFM-2K, demonstrating its superior recommendation accuracy and ranking quality. (3) Different compression methods significantly impact BFedRec's accuracy. The compression method used in BFedRec outperforms the other two compression competitors.

These experimental results indicate that BFedRec can achieve competitive performance while significantly reducing communication costs compared to common FL methods and compression methods.

TABLE 4 | Recommendation performance of different FL methods.

	MovieLens-1M		LastFM-2K	
Methods	**HR**	**NDCG**	**HR**	**NDCG**
Centralized	0.63	0.35	0.84	0.66
Independent	0.44	0.24	0.59	0.39
FedSGD	0.51	0.29	0.66	0.46
FedAvg	0.54	0.31	0.67	0.48
FedProx	0.55	0.31	0.68	0.48
MOON	0.57	0.33	0.71	0.50
SCAFFOLD	0.57	0.34	0.70	0.51
FedDyn	0.58	0.35	0.71	0.52
FedNCF	0.45	0.25	0.65	0.40
FedRecon	0.50	0.27	0.68	0.44
MetaMF	0.43	0.20	0.63	0.41
BFedRec-TopK	0.52	0.32	0.68	0.51
BFedRec-SVD	0.61	0.37	0.72	0.54
BFedRec	0.61	0.38	0.74	0.55

7.2.2 | Training Time

We conducted further experiments using two compression methods with compression rates comparable to BFedRec. For instance, in datasets LastFM-2K and MovieLens-1M, we fixed the user and item embeddings of the base model to 64. We then adjusted the compression rates for BFedRec and its variants using SVD and Top-K compression, ranging from {1/2, 1/4, 1/8, 1/16, 1/32, 1/64}. For fairness, the same compression rates are set for both uploading and downloading information. Table 5 lists our records of communication and computation times. (1) As the embedding size increases, both communication and computation times significantly increase. (2) In terms of communication, our method outperforms other compression methods, especially SVD compression. This is because our method reduces the uplink and downlink communication costs, leading to a more efficient training process. (3) In terms of computation, our method outperforms TopK compression, but is slightly inferior to SVD compression. This is because our method requires additional computation to update the low-rank structure.

In conclusion, BFedRec achieves a better balance between communication and computation costs, outperforming other BFedRec variants using SVD and Top-K compression. This is because BFedRec reduces the uplink and downlink communication costs, leading to a more efficient training process.

7.3 | Blockchain Performance Analysis for Latency and Scalability (RQ2)

7.3.1 | Latency

To evaluate the latency performance of our method in the blockchain network, we simulated BFedRec with varying numbers of blockchain nodes and different committee sizes. We recorded the time consumption in the four phases. As shown in Table 6, BFedRec is deployed on 200 and 100 nodes with different committee sizes, forming three parameter settings {(100,15), (100,30), (200,15)}. The values represent the average times for training 50 rounds. (1) Of all the phases, the voting phase requires the most time, and its duration increases twofold as the committee size expands due to the need for a greater number of "yes" votes to verify candidate blocks. (2) The times for both committee formation and the distribution of validated blocks are almost identical and relatively brief. (3) The time for committee formation increases with the committee size, doubling as the number of members rises, while the block propagation time doubles when the number of nodes in the blockchain grows.

TABLE 5 | Training time of different compression methods.

	MovieLens-1M			LastFM-2K		
Methods	**Comm.**	**Comp.**	**Total**	**Comm.**	**Comp.**	**Total**
Base	60.5	240.5	301.0	45.1	135.2	180.3
TopK@1/2	45.4	211.0	256.4	33.2	125.8	159.0
TopK@1/4	40.3	201.3	241.6	30.1	121.0	151.1
TopK@1/8	25.3	191.3	216.6	18.1	110.8	128.9
TopK@1/16	22.1	170.5	192.6	16.0	90.2	106.2
TopK@1/32	20.1	160.4	180.5	14.8	80.1	94.9
TopK@1/64	16.0	150.3	166.3	12.3	70.1	82.4
SVD@1/2	72.0	175.5	247.5	58.1	92.2	150.3
SVD@1/4	62.4	172.5	234.9	51.1	91.2	142.3
SVD@1/8	55.4	171.0	226.4	55.2	90.7	145.9
SVD@1/16	56.3	168.5	224.8	51.5	89.2	140.7
SVD@1/32	52.2	166.5	218.7	49.0	88.2	137.2
SVD@1/64	48.2	164.5	212.7	48.7	87.2	135.9
BFedRec@1/2	44.3	185.8	230.1	20.1	95.3	115.4
BFedRec@1/4	39.2	183.7	222.9	19.1	93.9	113.0
BFedRec@1/8	25.3	181.1	206.4	18.1	92.7	110.8
BFedRec@1/16	21.4	175.6	197.0	17.1	91.1	108.2
BFedRec@1/32	19.0	169.6	188.6	16.1	90.1	106.2
BFedRec@1/64	15.3	165.5	180.8	15.1	88.4	103.5

TABLE 6 | Time of different phases in blockchain network.

	MovieLens-1M			LastFM-2K		
Phases	**(100,15)**	**(100,30)**	**(200,15)**	**(100,15)**	**(100,30)**	**(200,15)**
Committee Constitution	0.1	0.2	0.1	0.1	0.2	0.1
Candidate Block Generation	6.0	5.9	6.0	5.7	5.6	5.9
Voting	91.4	149.3	95.9	85.5	133.8	87.9
Verified Block Propagation	0.3	0.6	0.6	0.3	0.6	0.6
Total Time	97.8	156.0	102.6	91.0	140.2	94.5

7.3.2 | Scalability

To evaluate the scalability of BFedRec, we conducted experiments by adjusting the committee size while maintaining a constant 100 blockchain nodes. In a separate setup, we changed the number of blockchain nodes while fixing the committee size at 15. The MovieLens-1M and LastFM-2K tasks were executed again under these varying configurations.

As illustrated in Figure 2, we observed two key trends: (1) The total time required to verify each block increases with the size of the committee, primarily due to the extended voting phase. While larger committee sizes ensure the availability of sufficient candidate members during the formation phase and improve the credibility of blocks through broader consensus, the increased communication overhead required to reach agreement on candidate blocks introduces significant delays. This trade-off highlights the balance between achieving robust consensus and maintaining efficiency in block verification. (2) Conversely, as the number of blockchain nodes grows, the time needed for finalizing and distributing blocks exhibits only a minimal increase. This efficiency is achieved by storing only the IPFS addresses of model data in the blockchain, rather than the full global or local model updates. Additionally, outdated model updates are pruned, reducing block size and facilitating faster transmission across the network. These optimizations significantly mitigate the potential performance degradation associated with larger node populations.

7.4 | Security Analysis for Resilience to Attacks and System Robustness (RQ3)

7.4.1 | Resistance to Poisoning Attacks

To test BFedRec's resistance to both data and model poisoning, we carried out experiments targeting two major attack vectors. Data

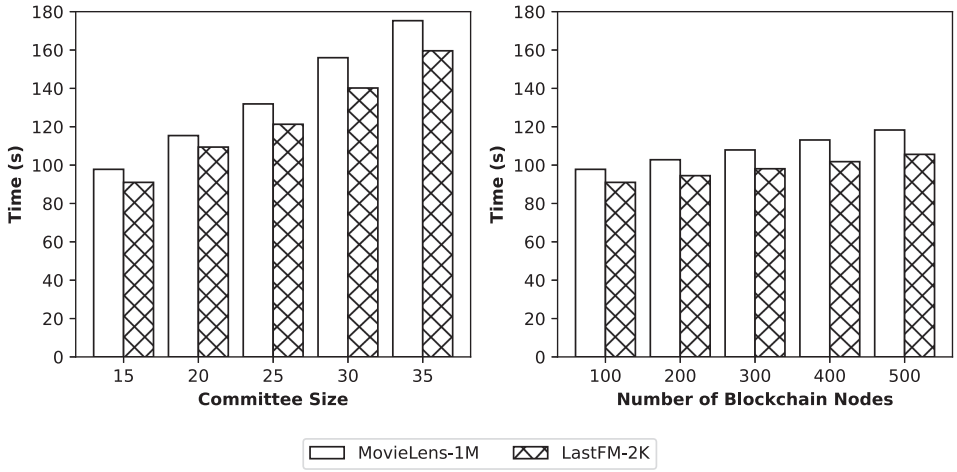

FIGURE 2 | Total time of BFedRec with difference committee sizes (left) and number of blockchain nodes (right).

poisoning takes place when adversarial clients inject corrupted or misleading entries during the dataset construction process. In federated learning, one frequent method of this is the Label Flipping (LF) attack [14], where attackers intentionally assign incorrect labels to certain data points. In contrast, model poisoning involves adversaries intentionally modifying their model updates to distort the overall learning process. A widely observed strategy is the Bit-Flipping (BF) attack [60], sometimes referred to as sign inversion, where manipulated gradients with reversed directions are submitted to the aggregation server.

We tested BF attacks across datasets MovieLens-1M and Lastfm-2K, and 20% of clients are set malicious. As illustrated in Figure 3, we can conclude: (1) BFedRec demonstrated a substantial improvement in performance under BF attacks compared to the standard FedAvg method. This result is evident because FedAvg does not employ any defense mechanisms. (2) BFedRec and Biscotti showed similar performance under BF attack scenarios, as both methods leverage blockchain technology to defend against malicious clients and enhance the security of model aggregation. However, BFedRec exhibited greater resilience across all datasets. Its HR and NDCG scores remained significantly higher, reflecting its ability to maintain superior recommendation quality and ranking accuracy even when a portion of the clients are compromised.

7.4.2 | Robustness Against Byzantine Nodes

The security of BFedRec is built upon a committee-based consensus protocol, designed to ensure robust and secure model aggregation. However, it is nearly impossible to fully validate its security by testing every possible attack vector. In a decentralized system like BFedRec, malicious blockchain nodes can attempt to disrupt the system in various ways, such as manipulating committee formation, tampering with candidate block generation, or skewing the voting process. These malicious nodes may cast dissenting votes against legitimate candidate blocks while approving incorrect ones, in an effort to corrupt the consensus mechanism.

A particularly insidious form of attack occurs during candidate block generation, where adversarial nodes launch Gaussian-style attacks by injecting arbitrary values into the global model parameters, rather than following the secure aggregation protocol. This attack attempts to poison the model by introducing random noise into the global model, disrupting the learning process and degrading system performance. To assess BFedRec's resilience to such adversarial behavior, we conducted an experiment where one-third of the blockchain nodes were deliberately made malicious. These malicious nodes colluded with one another in an attempt to mislead the learning process by voting against correct candidate blocks and injecting faulty model updates.

The results, illustrated in Figure 4, demonstrate that BFedRec maintained its performance despite the presence of these Byzantine nodes. Although the malicious nodes actively attempted to manipulate the system, BFedRec's consensus mechanism and secure aggregation protocol successfully mitigated their impact. The recommendation performance, as measured by metrics like Hit Rate (HR) and NDCG, showed only a minimal decline, proving the robustness of BFedRec in the face of such coordinated attacks.

In summary, the blockchain-assisted method of BFedRec provides stronger defenses against poisoning attacks and Byzantine attacks, offering significant advantages in terms of security and performance in federated recommendation systems (FedRSs).

8 | Conclusion

In this paper, we propose BFedRec, a blockchain-assisted federated edge learning method designed for recommendation systems. BFedRec leverages blockchain technology deployed on edge nodes to decentralize the aggregation and distribution of recommendation models, reducing reliance on central servers. To further optimize the process, a low-rank FL method is employed, significantly reducing communication costs and enhancing the security of aggregation and distribution in BFedRec. Experimental results demonstrate that BFedRec enhances communication efficiency and security while maintaining recommendation performance.

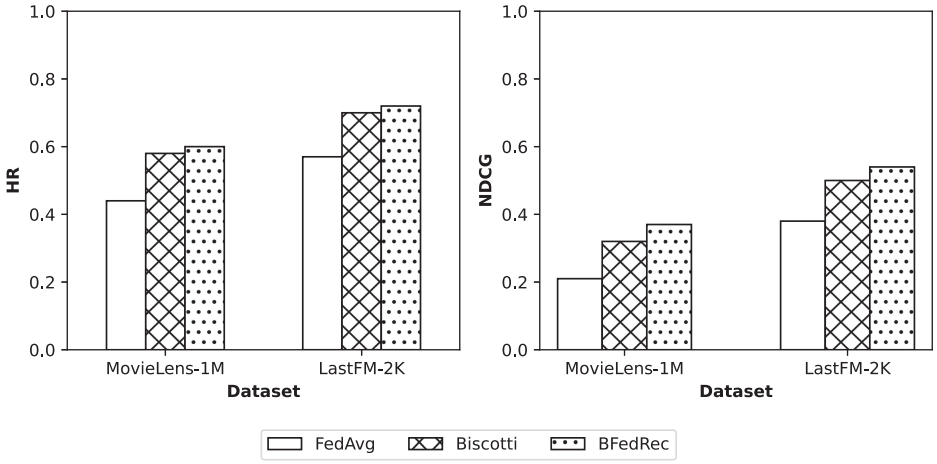

FIGURE 3 | Performance of FedRec vs. BFedRec with BF attacks.

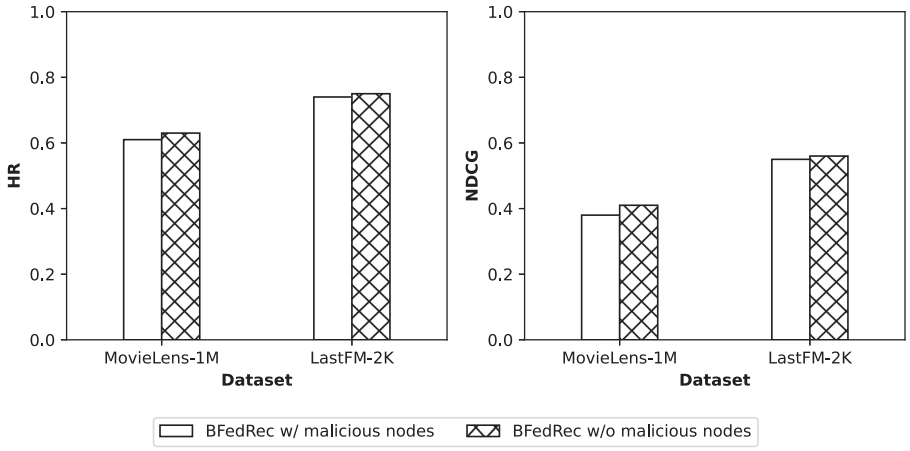

FIGURE 4 | Performance with vs. without malicious nodes.

The BFedRec framework is characterized by decentralization, security, and privacy protection, and it possesses excellent generality and extensibility, which allows it to be applied to a multitude of different scenarios. In an ongoing research project concerning trustworthy interactions between blockchain and off-chain entities, we will apply and extend the strategy of data aggregation and distribution based on blockchain, enabling the aggregation and distribution of data for distributed SDN controllers, which aids in enhancing the management efficiency and security of on-chain and off-chain communication networks.

Acknowledgments

The project was supported in part by the National Key R&D Program of China under Grant No. 2022YFB2702800 and the National Natural Science Foundation of China under Grant No. 92267104.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. Y. Gong, Z. Jiang, Y. Feng, et al., "Edgerec: Recommender System on Edge in Mobile Taobao," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Association for Computing Machinery, 2020), 2477–2484.

2. S. Wang, S. Guo, L. Wang, T. Liu, and H. Xu, "Hdnr: A Hyperbolic-Based Debiased Approach for Personalized News Recommendation," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, 2023), 259–268.

3. Y. Hu, X. Xu, L. Duan, M. Bilal, Q. Wang, and W. Dou, "End-Edge Collaborative Inference of Convolutional Fuzzy Neural Networks for Big Data-Driven Internet of Things," *IEEE Transactions on Fuzzy Systems* 33 (2024): 203–217.

4. X. Xu, H. Dong, L. Qi, et al., "Cmclrec: Cross-Modal Contrastive Learning for User Cold-Start Sequential Recommendation," in *Proceeding of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)* (Association for Computing Machinery, 2024).

5. S. L. Pardau, "The California Consumer Privacy Act: Towards a European-Style Privacy Regime in the United States," *Journal of Technology Law & Policy* 23 (2018): 68.

6. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks From

Decentralized Data," in *Artificial Intelligence and Statistics, PMLR* (PMLR, 2017), 1273–1282.

7. M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, et al., "Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System," (2019), arXiv preprint arXiv:1901.09888.

8. G. Lin, F. Liang, W. Pan, and Z. Ming, "Fedrec: Federated Recommendation With Explicit Feedback," *IEEE Intelligent Systems* 36, no. 5 (2020): 21–30.

9. M. Naumov, D. Mudigere, H.-J. M. Shi, et al., "Deep Learning Recommendation Model for Personalization and Recommendation Systems," (2019), arXiv preprint arXiv:1906.00091.

10. J. Yi, F. Wu, C. Wu, R. Liu, G. Sun, and X. Xie, "Efficient-Fedrec: Efficient Federated Learning Framework for Privacy-Preserving News Recommendation," (2021), arXiv preprint arXiv:2109.05446.

11. T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine* 37, no. 3 (2020): 50–60.

12. T. Vogels, S. P. Karimireddy, and M. Jaggi, "Powersgd: Practical Low-Rank Gradient Compression for Distributed Optimization," *Advances in Neural Information Processing Systems* 32 (2019).

13. P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning With Adversaries: Byzantine Tolerant Gradient Descent," *Advances in Neural Information Processing Systems* 30 (2017).

14. C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating Sybils in Federated Learning Poisoning," 2018.arXiv preprint arXiv:1808.04866.

15. D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates," in *International Conference on Machine Learning* (Pmlr, 2018), 5650–5659.

16. K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust Aggregation for Federated Learning," *IEEE Transactions on Signal Processing* 70 (2022): 1142–1154.

17. R. Guerraoui, and S. Rouault, "The Hidden Vulnerability of Distributed Learning in Byzantium," in *International Conference on Machine Learning* (PMLR, 2018), 3521–3530.

18. M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A Blockchain System for Private and Secure Federated Learning," *IEEE Transactions on Parallel and Distributed Systems* 32, no. 7 (2020): 1513–1525.

19. H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust Blockchained Federated Learning With Model Validation and Proof-Of-Stake Inspired Consensus," (2021), arXiv preprint arXiv:2101.03300.

20. M. Xu, Z. Zou, Y. Cheng, Q. Hu, D. Yu, and X. Cheng, "Spdl: A Blockchain-Enabled Secure and Privacy-Preserving Decentralized Learning System," *IEEE Transactions on Computers* 72, no. 2 (2022): 548–558.

21. Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A Blockchain-Based Decentralized Federated Learning Framework With Committee Consensus," *IEEE Network* 35, no. 1 (2020): 234–241.

22. S. Yuan, B. Cao, M. Peng, and Y. Sun, "Chainsfl: Blockchain-Driven Federated Learning From Design to Realization," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)* (IEEE, 2021), 1–6.

23. J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable Federated Learning for Mobile Networks," *IEEE Wireless Communications* 27, no. 2 (2020): 72–80.

24. J. Liang, S. Li, B. Cao, W. Jiang, and C. He, "Omnilytics: A Blockchain-Based Secure Data Market for Decentralized Machine Learning," (2021), arXiv preprint arXiv:2107.05252.

25. V. Mugunthan, R. Rahman, and L. Kagal, "Blockflow: An Accountable and Privacy-Preserving Solution for Federated Learning," 2020.arXiv preprint arXiv:2007.03856.

26. R. Jin, J. Hu, G. Min, and J. Mills, "Lightweight Blockchain-Empowered Secure and Efficient Federated Edge Learning," *IEEE Transactions on Computers* 72, no. 11 (2023): 3314–3325, https://doi.org/10.1109/TC.2023.3293731.

27. D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure Federated Matrix Factorization," *IEEE Intelligent Systems* 36, no. 5 (2020): 11–20.

28. Y. Lin, P. Ren, Z. Chen, et al., "Meta Matrix Factorization for Federated Rating Predictions," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, 2020), 981–990.

29. J. Wu, Q. Liu, Z. Huang, et al., "Hierarchical Personalized Federated Learning for User Modeling," in *Proceedings of the Web Conference* (Association for Computing Machinery, 2021), 957–968.

30. C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A Federated Graph Neural Network Framework for Privacy-Preserving Personalization," *Nature Communications* 13, no. 1 (2022): 3091.

31. Z. Ye, X. Zhang, X. Chen, H. Xiong, and D. Yu, "Adaptive Clustering Based Personalized Federated Learning Framework for Next Poi Recommendation With Location Noise," *IEEE Transactions on Knowledge and Data Engineering* 36, no. 5 (2024): 1843–1856, https://doi.org/10.1109/TKDE.2023.3312511.

32. X. He, S. Liu, J. Keung, and J. He, "Co-Clustering for Federated Recommender System," in *Proceedings of the ACM on Web Conference 2024* (Association for Computing Machinery, 2024), 3821–3832.

33. C. Zhang, G. Long, T. Zhou, Z. Zhang, P. Yan, and B. Yang, "When Federated Recommendation Meets Cold-Start Problem: Separating Item Attributes and User Interactions," in *Proceedings of the ACM on Web Conference 2024* (Association for Computing Machinery, 2024), 3632–3642.

34. N.-H. Nguyen, T.-A. Nguyen, T. Nguyen, V. T. Hoang, D. D. Le, and K.-S. Wong, "Towards Efficient Communication and Secure Federated Recommendation System via Low-Rank Training," in *Proceedings of the ACM on Web Conference 2024* (Association for Computing Machinery, 2024), 3940–3951.

35. L. Qu, W. Yuan, R. Zheng, L. Cui, Y. Shi, and H. Yin, "Towards Personalized Privacy: User-Governed Data Contribution for Federated Recommendation," 2024.arXiv preprint arXiv:2401.17630.

36. B. Yan, Y. Cao, H. Wang, W. Yang, J. Du, and C. Shi, "Federated Heterogeneous Graph Neural Network for Privacy-Preserving Recommendation," (2023), arXiv preprint arXiv:2310.11730.

37. H. Zhang, F. Luo, J. Wu, X. He, and Y. Li, "Lightfr: Lightweight Federated Recommendation With Privacy-Preserving Matrix Factorization," *ACM Transactions on Information Systems* 41, no. 4 (2023): 1–28.

38. F. K. Khan, A. Flanagan, K. E. Tan, Z. Alamgir, and M. Ammad-Ud-Din, "A Payload Optimization Method for Federated Recommender Systems," in *Proceedings of the 15th ACM Conference on Recommender Systems* (Association for Computing Machinery, 2021), 432–442.

39. V. Perifanis, G. Drosatos, G. Stamatelatos, and P. S. Efraimidis, "Fedpoirec: Privacy-Preserving Federated Poi Recommendation With Social Influence," *Information Sciences* 623 (2023): 767–790.

40. F. Liang, W. Pan, and Z. Ming, "Fedrec++: Lossless Federated Recommendation With Explicit Feedback," *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021): 4224–4231.

41. S. Liu, S. Xu, W. Yu, Z. Fu, Y. Zhang, and A. Marian, "Fedct: Federated Collaborative Transfer for Recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, 2021), 716–725.

42. C. Wu, F. Wu, T. Qi, Y. Huang, and X. Xie, "Fedattack: Effective and Covert Poisoning Attack on Federated Recommendation via Hard Sampling," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge*

Discovery and Data Mining (Association for Computing Machinery, 2022), 4164–4172.

43. W. Yuan, Q. V. H. Nguyen, T. He, L. Chen, and H. Yin, "Manipulating Federated Recommender Systems: Poisoning With Synthetic Users and Its Countermeasures," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, 2023), 1690–1699.

44. S. Zhang, H. Yin, T. Chen, Z. Huang, Q. V. H. Nguyen, and L. Cui, "Pipattack: Poisoning Federated Recommender Systems for Manipulating Item Promotion," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Association for Computing Machinery, 2022), 1415–1423.

45. Y. Qu, L. Gao, T. H. Luan, et al., "Decentralized Privacy Using Blockchain-Enabled Federated Learning in Fog Computing," *IEEE Internet of Things Journal* 7, no. 6 (2020): 5171–5183.

46. Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial Iot," *IEEE Transactions on Industrial Informatics* 16, no. 6 (2019): 4177–4186.

47. W. Ali, X. Zhou, and J. Shao, "Privacy-Preserved and Responsible Recommenders: From Conventional Defense to Federated Learning and Blockchain," *ACM Computing Surveys* 57, no. 5 (2025): 1–35.

48. W. Ali, R. Kumar, X. Zhou, and J. Shao, "Responsible Recommendation Services With Blockchain Empowered Asynchronous Federated Learning," *ACM Transactions on Intelligent Systems and Technology* 15, no. 4 (2024): 1–24.

49. P. Chatterjee, D. Das, and D. B. Rawat, "Federated Learning Empowered Recommendation Model for Financial Consumer Services," *IEEE Transactions on Consumer Electronics* 70, no. 1 (2023): 2508–2516.

50. B. B. Gupta, A. Gaurav, and V. Arya, "Secure and Privacy-Preserving Decentralized Federated Learning for Personalized Recommendations in Consumer Electronics Using Blockchain and Homomorphic Encryption," *IEEE Transactions on Consumer Electronics* 70, no. 1 (2023): 2546–2556.

51. P. Xiao, S. Cheng, V. Stankovic, and D. Vukobratovic, "Averaging Is Probably Not the Optimum Way of Aggregating Parameters in Federated Learning," *Entropy* 22, no. 3 (2020): 314.

52. F. M. Harper and J. A. Konstan, "The Movielens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems* 5, no. 4 (2015): 1–19.

53. I. Cantador, P. Brusilovsky, and T. Kuflik, "Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (hetrec2011)," in *Proceedings of the Fifth ACM Conference on Recommender Systems* (Association for Computing Machinery, 2011), 387–388.

54. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," in *Proceedings of the 26th International Conference on World Wide Web* (International World Wide Web Conferences Steering Committee, 2017), 173–182.

55. Q. Li, B. He, and D. Song, "Model-Contrastive Federated Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers, 2021), 10713–10722.

56. S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic Controlled Averaging for On-Device Federated Learning," 2 (6), (2019), arXiv preprint arXiv:1910.06378.

57. D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated Learning Based on Dynamic Regularization," 2021.arXiv preprint arXiv:2111.04263.

58. V. Perifanis and P. S. Efraimidis, "Federated Neural Collaborative Filtering," *Knowledge-Based Systems* 242 (2022): 108441.

59. K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated Reconstruction: Partially Local Federated Learning," *Advances in Neural Information Processing Systems* 34 (2021): 11220–11232.

60. S. P. Karimireddy, L. He, and M. Jaggi, "Learning From History for Byzantine Robust Optimization," in *International Conference on Machine Learning* (PMLR, 2021), 5311–5319.