

Blockchain-based Federated Learning: A Systematic Survey

Junqin Huang, Linghe Kong, *Senior Member, IEEE,* Guihai Chen, Qiao Xiang, Xi Chen, and Xue Liu, *Fellow, IEEE*

***Abstract*—Federated Learning (FL) is a distributed machine learning paradigm that trains models across multiple devices without exchanging users' data, thereby providing stronger data privacy guarantees. However, some research reveals that FL may face security and privacy issues, such as single point of failure, model poisoning, and parameter privacy disclosure. Recently, the field of combining blockchain and FL has trend to become a hot research topic. More and more researchers attempt to use blockchain to decentralize and secure FL frameworks. To further understand recent advances in blockchain-based FL (BFL) systems, this article aims to provide a systematic survey to deconstruct BFL systems. We propose a taxonomy of BFL systems following the lifecycle of FL tasks and divide them into three layers, *i.e.*, the blockchain layer, the training layer, and the aggregation layer. We review and summarize representative work in each layer. We also discuss several open challenges for designing more secure and efficient BFL systems.**

***Index Terms*—Blockchain, Federated Learning.**

I. Introduction

Federated Learning (FL), proposed by Google in 2016, has been recognized as a promising technique to address privacy concerns in machine learning algorithms. One classic case is that Google uses the FL technique to help Gboard learn new words and phrases without collecting the text users speak or type. To enable the distributed training paradigm, FL distributes models across multiple users' devices (*i.e.*, workers) and trains with their local data. Workers transfer their updated model parameters to the parameter server (*i.e.*, aggregator) for aggregating the global model. Therefore, the global model can be updated without knowing local training data. Since workers do not expose their local data during training, people once considered that the FL technique could prevent data disclosure well. However, there remain some security and privacy issues [1] to be addressed:

- *Data privacy*. Even though workers' data are not exposed directly in the training process, recent advances show that adversaries could reconstruct raw data in light of updated model parameters.
- *Model poisoning*. Adversaries could launch data poisoning or model poisoning attacks to compromise the model performance, such as training local models using anomalous data and modifying updated model parameters.
- *Centralization*. FL systems that rely on centralized aggregators are vulnerable to single-point of failures and DDoS attacks. Besides, the system performance is constrained by the limited bandwidth and computation powers of the aggregator.
- *Opaque incentives*. Incentive mechanisms are also important in FL systems. They provide proper rewards to workers who make positive contributions to training models. Fair incentive mechanisms help raise workers' enthusiasm to participate in FL tasks. But a centralized, opaque system might provide unfair incentives (*e.g.*, Amazon MTurk [2]).

Moreover, FL systems usually have a strong assumption that aggregators are always honest, which is not practical in the real world. To avoid that system reliability depends only on a single node, it is a natural thought to build an FL system with a decentralized paradigm for making the system more secure and robust.

With the emergence of blockchain technologies, it is promising to address the aforementioned security and privacy issues in FL. Thanks to the decentralized architecture of blockchain, blockchain-based FL (BFL) systems do not need to rely on a centralized aggregator but a decentralized consensus to aggregate updated model parameters. Specifically, blockchain nodes in BFL systems reach a consensus on their local training results. Under the premise that the majority of nodes are honest, such a decentralized aggregation paradigm can alleviate malicious attacks, such as poisoning attacks, DDoS attacks, and single-point of failures. By canceling the centralized aggregator, BFL systems are more robust and scalable. Moreover, the immutability and traceability features of blockchain ensure BFL systems record all updated model parameters during the training process faithfully and guarantee the recorded parameters are not tampered with and the training process is auditable. In addition, smart contracts running on top of blockchain can provide workers with transparent, customized, and undeniable incentive mechanisms. A fair and proper incentive mechanism can motivate workers to participate in the BFL systems well.

Benefiting from the superiority of blockchain, more and more researchers show great interest in the field of combining blockchain and FL. However, research in this field is still in its infancy. Up to now, there only exists one related survey investigating the combination of blockchain and FL in edge computing [3], which mainly focuses on classifying recent advances from issues and application aspects, *e.g.*, data sharing, content caching, and crowdsensing. A systematic survey toward deconstructing the design components of existing BFL systems is still missing, which can help researchers understand how to design a practical BFL system. To fill this gap, we conduct a brief but thorough survey by identifying and classifying the most high-quality literature, which is closely related to BFL systems.

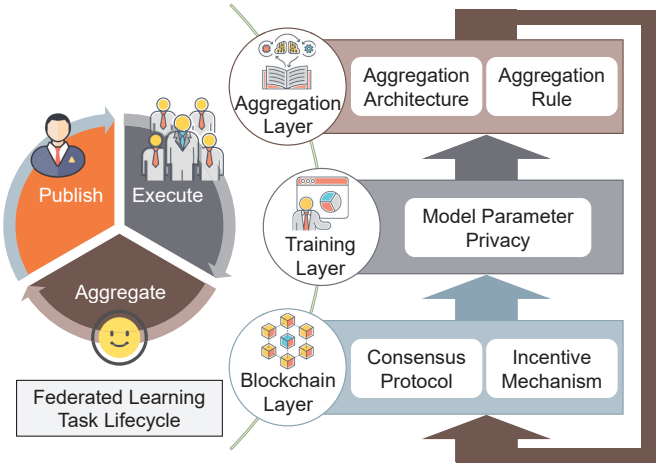

Fig. 1. Taxonomy of BFL systems corresponding to the FL task lifecycle.

II. Taxonomy

In this article, we propose a taxonomy of BFL systems following the lifecycle of an FL task, as shown in Figure 1. We deconstruct BFL systems into three layers from bottom to up: *blockchain layer*, *training layer*, and *aggregation layer*, corresponding to the three phases in an FL task lifecycle.

In the *publish* phase, a requester publishes its FL task via a smart contract to the blockchain. The FL task contract generally contains information like an initial task model, a reward scheme, and training criteria. Since the publication of an FL task contract should be confirmed by blockchain nodes, we have to design a suitable consensus protocol according to different system scales and permission control strategies. When formulating FL tasks in smart contracts, requesters need to devise proper incentive mechanisms for distributing task rewards fairly and transparently. After the global model is aggregated and checked, workers will be rewarded automatically according to the incentive mechanism predefined in the smart contract. Thus, we assign the design of consensus protocol and incentive mechanism to the *blockchain layer*.

In the *execute* phase, workers receive published FL tasks from the blockchain by triggering corresponding functions in the smart contract and train local models using their data. During training models, workers should protect their data privacy from leaking through model parameters [1]. Therefore, BFL systems should introduce data privacy protection schemes for workers in the *training layer*.

In the *aggregate* phase, workers submit their updated model parameters to the aggregator or blockchain nodes to aggregate the global model, where the aggregation architecture and aggregation rule should be considered. The aggregation architecture determines the way to gather local model parameters. Different architectures have different communication complexity, which further impacts system scalability. And the aggregation rule helps figure out anomalous parameters and improve the quality of global models. We include these two critical components in the *aggregation layer*.

In the rest of this article, we will look back at the existing enabling techniques for BFL systems in the proposed three layers, respectively.

III. Blockchain Layer

In the blockchain layer, we focus on the design of consensus protocols and incentive mechanisms for BFL systems.

A. Consensus Protocol

The consensus protocol is one of the most important components of a blockchain. Blockchain nodes should reach a consensus on all blockchain data to prevent Byzantine nodes from doing evil. Since different consensus protocols have different scalability, safety, and convergence rate, we have to design suitable consensus protocols according to the requirement of BFL systems. There are mainly three types of consensus protocols applied in existing BFL systems: *probabilistic consensus*, *deterministic consensus*, and *hybrid consensus*, which are summarized in Table I.

1) Probabilistic Consensus: The probabilistic consensus is like a lottery game, where everyone has an equal probability of winning this game. Blockchain nodes (*i.e.*, miners) who win the game can mine a new block successfully. Proof-of-work (PoW) [4] is one of the most popular probabilistic consensus adopted in Bitcoin and Ethereum. In PoW, miners use the hash function to generate random outputs constantly. The miner who first generates an output value smaller than the mining difficulty value obtains the right to mine a new block. Due to the randomness of the hash function, miners cannot construct a dedicated input for the targeted output value, which guarantees the fairness of PoW. However, PoW wastes too many resources in hash calculation. Thus, proof-of-stake (PoS) [5], [6] was proposed to relieve this phenomenon. PoS evolves from PoW but additionally brings the concept of *stake age* to relax the mining difficulty, which helps accelerate the mining process and reduce resource consumption.

Some BFL systems proposed proof-of-useful-work (PoUW) consensus protocols to replace "useless" hash calculation in PoW/PoS. For example, FedCoin [7] designed the proof-of-Shapley (PoSap) consensus, which makes miners compute Shapley values to assess workers' contribution to FL tasks while satisfying the requirement of blockchain consensus. BytoChain [8] and BFDS [9] combine the training and consensus process, and workers having the highest training model accuracy will obtain the right to generate new blocks. However, the security of these newly proposed PoUW consensus protocols has not been fully proven.

Probabilistic consensus protocols are highly scalable due to their relatively low communication complexity and are usually adopted in large-scale public blockchains. But due to the low convergence rate, they cannot reach a consensus quickly.

2) Deterministic Consensus: Deterministic consensus protocols can reach a consensus in fixed steps, so they have higher throughput than probabilistic consensus protocols. Practical Byzantine fault tolerance (PBFT) is a representative deterministic consensus protocol [10], [11]. PBFT usually takes four steps to achieve consensus, *i.e.*, *pre-prepare*, *prepare*, *commit*, and *reply*. During this process, blockchain nodes need to exchange messages with each other, so the communication

TABLE I
SUMMARY OF BFL SYSTEMS IN THE BLOCKCHAIN LAYER (COMM. COMP.=COMMUNICATION COMPLEXITY; n=# OF CONSENSUS NODES; f=# OF BYZANTINE NODES; c=# OF COMMITTEE MEMBERS).

Ref.	Consensus		Blockchain	Safety	Scalability	Comm. Comp.	Incentive Mechanism
LearningChain [4]	Probabilistic Consensus	PoW	Ethereum	$n \geq 2f+1$	High	$O(n)$	-
SecCL [5]		PoW/PoS	Ethereum	$n \geq 2f+1$	High	$O(n)$	-
SFL [6]		PoW/PoS	Ethereum	$n \geq 2f+1$	High	$O(n)$	-
FedCoin [7]		PoSap	-	$n \geq 2f+1$	High	$O(n)$	Shapley Value
BytoChain [8]		PoA	-	$n \geq 2f+1$	High	$O(n)$	Accuracy Increment
BFDS [9]		PoQ	-	$n \geq 2f+1$	High	$O(n)$	-
RFL [10]	Deterministic Consensus	PBFT	Hyperledger Fabric	$n \geq 3f+1$	Low	$O(n^2)$	-
BFLC [11]		PBFT	FISCO BCOS	$n \geq 3f+1$	Low	$O(n^2)$	Contribution-based
PIRATE [12]		HotStuff	-	$n \geq 3f+1$	Medium	$O(n)$	-
PBFL [13]	Hybrid Consensus	Algorand	-	$n \geq 3f+1$	Medium	$O(n \cdot c + c^2)$	Reputation-based
DeepChain [14]		Algorand	Corda	$n \geq 3f+1$	Medium	$O(n \cdot c + c^2)$	Contribution-based
Biscotti [15]		PoF	-	$n \geq 3f+1$	Medium	$O(n \cdot c + c^2)$	-

complexity of PBFT achieves $O(n^2)$, which is unbearable when the system scale becomes large. HotStuff [12] is an optimized variant of PBFT, which leverages the threshold signature technique to realize a linear communication complexity $O(n)$. Moreover, PIRATE [12] makes the HotStuff protocol pipelined to further enhance the consensus efficiency. Deterministic consensus protocols are more suitable for permissioned blockchains, where blockchain nodes must be authorized before joining the system.

3) Hybrid Consensus: The hybrid consensus combines the above two basic consensus types to make the trade-off between scalability and convergence rate. In general, the workflow of hybrid consensus protocols can be illustrated in two steps: First, utilize probabilistic consensus protocols to select c blockchain nodes to form the consensus committee. Second, apply deterministic consensus protocols to achieve consensus within the consensus committee. In this manner, the hybrid consensus protocols can support a large-scale blockchain system while improving the convergence rate compared to probabilistic consensus protocols. A classic hybrid consensus protocol is Algorand [13], [14]. It first utilizes the verifiable random function (VRF) to select a subset of blockchain nodes as the committee members in a non-interactive way and then uses a BFT-like protocol to achieve consensus within the committee. Biscotti [15] proposed a hybrid consensus protocol similar to Algorand, proof-of-federation (PoF), which leverages workers' contribution instead of stake as the input of VRF to select committee members.

B. Incentive Mechanism

Incentive mechanisms can motivate workers to engage in the FL tasks actively. A fair incentive mechanism should reward honest and high-quality workers, and punish or give no reward to malicious or low-quality workers. For a traditional centralized FL platform, the central server may provide an opaque, biased incentive mechanism [2]. In comparison, the blockchain smart contract provides a transparent and tamper-proof platform to customize incentive mechanisms. Workers can audit the incentive mechanism scheme formulated in the smart contract and verify its execution results. Most existing BFL systems adopt contribution-based or reputation-based reward and punishment mechanisms as their incentive mechanisms.

In BFLC [11], the committee members validate the local updates by treating their local data as a validation set, and the validation accuracy becomes the score. After the aggregation of each round, the committee members distribute rewards to the corresponding workers based on scores of their submitted updates. As a result, frequently providing updates could earn more rewards, and the constantly updated global model will attract more workers to participate. To fairly reward data holders (*i.e.*, workers), BytoChain [8] uses accuracy increment to quantify the contribution of data holders. The number of rewards is directly proportional to the increase in accuracy, where the accuracy is tamper-proof on the blockchain, and all workers can easily verify its increment. Therefore, workers will lose the motivation to launch the free-riding attack because the accuracy increments of their useless models are close to zero, so they can no longer get rewards.

Similarly, PBFL [13] and DeepChain [14] reward workers based on their reputation/contributions. By combining Multi-Krum and the reputation-based scheme, PBFL can reward workers properly and prevent poisoning attacks. Besides these intuitive contribution-based distribution schemes, we have the chance to further refine the reward distribution through game theory, mathematical optimization methods, etc. For example, FedCoin [7] adopts Shapley value in the game theory to calculate workers' contributions fairly and quantitatively, thereby incentivizing workers to contribute their efforts.

IV. TRAINING LAYER

Adversaries may infer the original data from the updated model parameters, which leaks workers' training data privacy [1]. Therefore, we focus on protecting workers' model parameter privacy in the training layer.

A. Model Parameter Privacy

Several attacks could infer the training data referring to the updated model parameters, such as membership inference, model inversion, and GAN reconstruction [1]. We divide the parameter privacy-preserving schemes used in existing BFL

TABLE II
SUMMARY OF BFL SYSTEMS IN THE TRAINING LAYER.

Ref.	Parameter Privacy-preserving Scheme	
LearningChain [4]	Noise Perturbation	LDP
Biscotti [15]		LDP
PBFL [13]		LDP
SFL [6]		LDP
BFDS [9]		LDP
DeepChain [14]	Private Aggregation	Threshold Paillier
Biscotti [15]		Shamir Secrets Sharing

systems into two categories: *noise perturbation* and *private aggregation*, which are summarized in Table II.

1) Noise Perturbation: The key idea of noise perturbation-based schemes is to add private noise to the local model parameters before sending them to others. Due to noise perturbation, attackers cannot infer the original data, even if they intercept local model parameters. A common perturbation technique is the differential privacy (DP) [1], which ensures that any response to queries is essentially equally likely to happen, regardless of whether a particular record is contained in the dataset. The "essentially" is measured by the parameter ε, and a smaller value indicates better privacy. If one function satisfies the above property, we can say it achieves ε-DP. However, the traditional DP technique needs a trusted third party to collect original data and add noise to the computation results. Since the global model is built by multiple parties, the local DP (LDP) technique is more suitable for BFL systems [6], [9], [13]. The updated model parameters are perturbed locally before release, so LDP does not reveal original data to any third party.

In SFL [6], workers use the Gaussian noise function to add noise to their local model updates before submitting them to the blockchain. This function satisfies the requirement of (ε, δ)-LDP, where δ is the probability that plain ε-LDP is broken. Workers use LDP to defend against membership inference attacks. Similarly, BFDS [9] and PBFL [13] add the Laplace noise to perturb local model parameters. However, noise perturbation-based schemes may decrease the performance of models, and it is hard to make the trade-off between privacy and accuracy, by tuning the parameter ε.

2) Private Aggregation: Another type of parameter privacy-preserving scheme utilizes cryptographic tools to implement private aggregation. DeepChain [14] applies the threshold Paillier algorithm that provides additive homomorphic property to encrypt gradients. It assumes a trusted setup is needed, and the secret key cannot leak without at least t of n workers. Workers can aggregate the encrypted model parameters from others due to the additive homomorphic property of cipher text. However, DeepChain cannot apply aggregation rules because the encrypted model parameters only support addition operations.

To enable robust aggregation rules in private aggregation, Biscotti [15] uses both noise perturbation and private aggregation methods. First, it masks the workers' model parameters using LDP and leverages the verification committee to validate these masked parameters. If the parameters pass the Multi-Krum verification, the committee will sign on the unmasked parameters. Second, it uses Shamir secret sharing, an additively homomorphic encryption scheme, for private aggregation. Since anomalous model updates are excluded in the first step, Biscotti can ensure the quality of the aggregated global model. Even though private aggregation can well protect parameter privacy and does not sacrifice model performance, these cryptographic methods are usually computation-intensive and unpractical.

V. AGGREGATION LAYER

After workers submit their updated model parameters to the network, these parameters should be checked and aggregated. Therefore, in the aggregation layer, we focus on the design of aggregation architectures and aggregation rules.

A. Aggregation Architecture

The aggregation architectures of existing BFL systems contain three basic types: *Parameter Server (PS)*, *All-Reduce*, and *Gossip*, which are shown in Figure 2 and summarized in Table III.

1) Parameter Server Architecture: The PS architecture is often adopted in the central aggregator-existed BFL systems. In this architecture, blockchain does not participate in the model aggregation but is taken as a trusted distributed ledger to record, transfer, and audit the updated model parameters or workers' behaviors. The PS architecture implicitly assumes that the central aggregator (*i.e.*, parameter server) is honest, so it cannot prevent attacks from the aggregator. As shown in Figure 2, we can see that there are three roles in the PS architecture: *central aggregator*, *workers*, and *blockchain nodes*. The central aggregator is a cloud server responsible for aggregating local models obtained from the blockchain. Workers train sub-models using their local datasets and store the updated model parameters on the blockchain. Blockchain nodes receive the local model parameters from workers, pack them as blockchain data, and synchronize them among all blockchain nodes. Blockchain nodes keep the consistency of blockchain data across the whole network by consensus algorithms. The blockchain serves as a service to provide a transparent, traceable, unforgeable intermediary to help exchange information between the aggregator and workers faithfully.

In SFL [6], the central aggregator publishes the FL task as a smart contract, which includes the initial model, encrypted testing data, the training criteria, etc. Workers download the initial model and train local models with their datasets. The updated model parameters will be submitted to the blockchain and evaluated using testing data in the smart contract. If the model parameters satisfy the criteria, *i.e.*, model accuracy, they will be accepted by the central aggregator for further aggregation. The encrypted testing data stored in the smart contract will be revealed after collecting all updated local models to prevent testing data disclosure. SecCL [5] utilizes the blockchain to store received parameters. Workers evaluate the parameters recorded in the blockchain and send evaluation scores to the aggregator. Only those high-score parameters can be used to generate the global model.

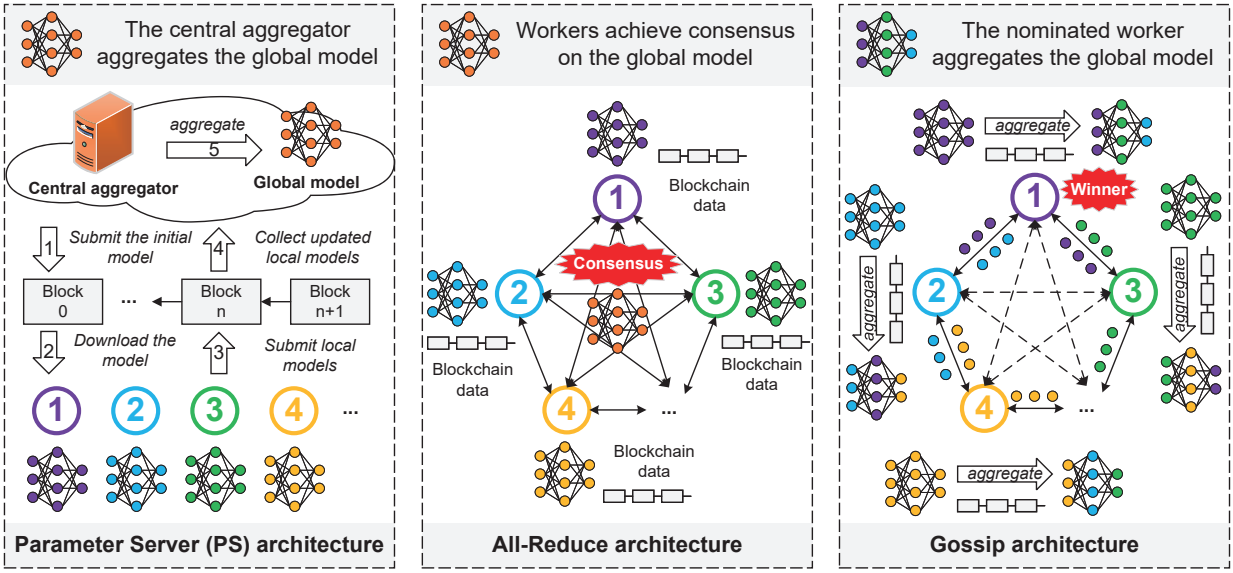

Fig. 2. Three basic model aggregation architectures of BFL systems.

Apart from utilizing the blockchain to store model parameters, some other research uses the blockchain to record workers' reputations or contribution [7], [10]. The central aggregator will select task candidates or distribute rewards according to their reputation or contribution recorded in the blockchain. Since model parameters are aggregated by the central aggregator, the PS architecture has a high model consistency. However, the scalability of PS architecture is limited to the bandwidth and computing capability of the central aggregator. And the PS architecture does not make full use of the distributed advantages of the blockchain.

2) All-Reduce Architecture: Compared to the PS architecture, the All-Reduce architecture-based BFL systems are fully decentralized and have higher scalability. Each node is equal in this architecture. Specifically, a node can be the worker in the model training stage, the aggregator in the model aggregation stage, and the blockchain node in the consensus stage. All the global results achieve consensus among the majority of nodes. As shown in Figure 2, workers first train local models using their data and broadcast their updated model parameters to other workers through the blockchain. After workers receive the model parameters from other workers, they can aggregate the global model locally. Thus, we can complete FL tasks without a central aggregator by utilizing the All-Reduce architecture. However, this architecture suffers from high communication complexity, *i.e.*, $O(n^2)$. Therefore, no existing BFL system directly uses this architecture but leverages some hierarchical or hybrid All-Reduce architectures to decrease the communication complexity, thereby improving the system's scalability.

Hierarchical All-Reduce. For example, PIRATE [12] divides workers into several committees and achieves consensus hierarchically from intra-committees to inter-committees. Specifically, workers first reach a consensus on the updated model parameters inside the committees using HotStuff. And then, committees exchange their locally-agreed aggregation results with their neighbors using the Ring All-Reduce manner. However, such an architecture has a potential threat that workers in different committees can hardly obtain models or records from other committees. If a committee as a whole is malicious, it is difficult for other honest committees to detect and resist.

Hybrid All-Reduce. To eliminate the possible attacks from malicious committees and have moderate communication complexity, several research [11], [13]–[15] proposed the hybrid All-Reduce architecture. Similar to the idea of hybrid consensus, the hybrid All-Reduce architecture firstly selects a certain number of workers to form a committee and then leverages the All-Reduce architecture to aggregate the global model within the committee. Since the updated model parameters are only validated and aggregated by a small set of workers, the scalability of the hybrid All-Reduce architecture is still acceptable.

3) Gossip Architecture: The Gossip architecture-based BFL systems do not need a central aggregator, either. The main difference between the Gossip architecture and the All-Reduce architecture is how the workers are connected: workers are fully connected in the All-Reduce architecture, while workers only connect with a few neighbors in the Gossip architecture. Workers only need to broadcast the updated model parameters to their neighbors, which significantly reduces communication complexity compared to the All-Reduce. However, the Gossip architecture cannot guarantee that each worker has the same data view, resulting in inconsistent global models. As shown in Figure 2, workers exchange updated model parameters with their neighbors and aggregate their final models locally. Since their received updated parameters are different, their final models are not the same. Thus, they will select a nominated worker through probabilistic consensus protocols (*e.g.*, PoW, PoS) to aggregate the final model [4]. The final model generated by the nominated worker will be taken as the global model. We observe that the global model is made up of

TABLE III
SUMMARY OF BFL SYSTEMS IN THE AGGREGATION LAYER.

Ref.	Aggregation Architecture	Scalability	Model Consistency	Aggregation Rule	
FedCoin [7]	Parameter Server	Medium	High	-	
SecCL [5]		Medium	High	Detection-based	PSF
SFL [6]		Low	High		Model Accuracy
RFL [10]		Medium	High		Reputation
PIRATE [12]	Hierarchical All-Reduce	High	High		Credit Score
BFLC [11]	Hybrid All-Reduce	Medium	Medium		Model Accuracy
DeepChain [14]		Medium	Medium	-	
Biscotti [15]		Medium	Medium	Tolerance-based	Multi-Krum
PBFL [13]		Medium	Medium		Multi-Krum
LearningChain [4]	Gossip	High	Low		l-nearest gradients
BFDS [9]		High	Low	Detection-based	Model Accuracy
BytoChain [8]		High	Low		AOT&ADT

a subset of updated parameters because the nominated worker may not receive complete updates from all workers. Thus, the model consistency of the Gossip architecture is low.

However, nominating workers via PoW/PoS does not guarantee the quality of the global model. If the nominated worker is malicious, it may generate the global model by choosing the wrong parameters. To solve the above issues, BytoChain [8] designed the proof-of-accuracy (PoA) consensus protocol, which selects the worker whose final model has the best performance to generate the global model. Similarly, BFDS [9] proposed the proof-of-training-quality (PoQ) consensus protocol to train the high-quality global model.

Putting Section V-A and Section III-A together, we can see that the blockchain consensus and aggregation architecture are highly correlated in the BFL systems. The All-Reduce architecture generally corresponds to deterministic consensus, while the Gossip architecture usually adopts probabilistic consensus. Different architectures have different scalability, convergence speed, and model consistency. Therefore, we need to flexibly design the system architecture according to the needs of specific scenarios.

B. Aggregation Rule

Aggregation architectures determine the system scalability and model consistency, while aggregation rules determine the model performance and security. In BFL systems, adversaries may be disguised as honest workers to disrupt the process of FL tasks, *i.e.*, launching data poisoning or model poisoning attacks [1]. To defend against these attacks, researchers design robust aggregation rules in BFL systems to realize Byzantine-robust FL. We divide them into two categories: *tolerance-based* and *detection-based* aggregation rules, which are summarized in Table III.

1) Tolerance-based Aggregation Rule: There are some commonly used tolerance-based aggregation rules to realize robust aggregation, such as Krum, Bulyan, trimmed mean, and median [1]. These rules have the similar intuitive idea that finding the most "center" models as the global model.

For example, Krum selects the local model closest to others as the global model. The distance between two models is measured by the Euclidean distance. However, Krum could be compromised by abnormal model parameters. To make Krum more robust, Multi-Krum was proposed [15], which selects multiple local models closest to other models, and averages these model parameters as the global model. Similarly, LearningChain [4] proposed an l-nearest gradients aggregation rule, which selects the top l closest gradients based on their cosine distances, and aggregates the global model using these l gradients.

However, all these tolerance-based aggregation rules face the same problem: If a Byzantine worker obtains the local model parameters of most workers, it can generate fake model parameters to launch an attack that arbitrarily alters the global model. Therefore, tolerance-based aggregation rules seem more of a reactive defense strategy.

2) Detection-based Aggregation Rule: Compared to tolerance-based aggregation rules, detection-based rules are a more active defense strategy that validates received local models and proactively eliminates anomalous ones. In general, detection-based rules measure the quality of local models by evaluating the model accuracy on testing data and discarding the unqualified local models [6], [9], [11].

For example, SecCL [5] proposed the positional scoring function (PSF) for selecting optimal model parameters to update the global model. In SecCL, workers will evaluate the updated models from other workers using their local data and send the evaluation scores to the aggregator, who selects and aggregates top local models according to the received evaluation scores. Therefore, SecCL can ensure only the high-quality local models can be chosen to update the global model. BytoChain [8] considered that selecting the local models only by accuracy may cause the over-fitting problem, so it proposed two thresholds, the accuracy oscillation threshold (AOT) and the accuracy deviation threshold (ADT), which can tolerate local models with a certain drop in accuracy. Because accuracy oscillations are common in the FL training, these models may help jump out of local optima.

Instead of directly validating the accuracy of local models, in PIRATE [12], the committees will assign corresponding credit scores for workers by considering multiple factors, such as computing capability, historical records, and network conditions. They use the credit scores to reconfigure the candidates of FL tasks and weight the received model parameters for robust aggregation. That is to say, workers whose credit scores

are higher will contribute more to the global model. Similarly, RFL [10] uses the reject on negative influence (RONI) scheme and the FoolsGold scheme to evaluate the quality of updates, denoted as workers' reputation.

Due to the decentralized advantage of the blockchain, BFL systems can well apply detection-based aggregation rules to exclude anomalous updates. Combining the validation results from other workers, BFL can achieve k-fold cross-validation using multiple workers' local data for robust aggregation.

VI. Challenges and Discussion

A. Challenges

BFL brings a new decentralized paradigm to traditional FL, which enables the aggregator-free FL, tamper-proof parameters record, transparent incentives, etc. However, there remain several open challenges that need to be addressed:

- *Heterogeneity.* In BFL systems, workers' computational power, network conditions, and training data are heterogeneous, which may largely affect the convergence rate of FL tasks, and even fail to converge. However, few BFL systems take them into account to avoid negative impacts, while this is precisely a critical challenge that hinders the deployment of BFL systems in practice.
- *Communication complexity.* Since the decentralized aggregation architecture has a higher communication complexity than the centralized PS architecture, how to improve the communication efficiency of BFL systems is another major challenge. Besides designing new aggregation architectures and blockchain consensus protocols, compressing or sparsing model parameters is another possible way to optimize communication efficiency.
- *Model security.* Existing poisoning defense methods may fail to prevent some local model poisoning attacks [1] and backdoor poisoning attacks. One novel poisoning defense method utilizes deep learning to aggregate gradients from workers [12], while it only suits the PS architecture having the central aggregator. How to defend against backdoor poisoning attacks in aggregator-free BFL systems is still unresolved.
- *Data privacy.* Private aggregation using cryptographic tools can protect parameter privacy well, but it is usually computation-heavy and only supports limited arithmetic operations. Noise perturbation, such as adding noise, gradient compression, and sparsification, is more lightweight, but it has a trade-off between model performance and data privacy. Some newly proposed methods leverage the trusted hardware (*e.g.*, Intel SGX) to do confidential computing, while the trusted hardware has a risk of side-channel attacks. Thus, it is non-trivial to design a suitable parameter privacy-preserving scheme for BFL systems.

B. Discussion

- *How to choose a proper aggregation architecture?* The PS architecture is more suitable for the scenarios where a central aggregator is needed, while the All-Reduce and Gossip architectures are both aggregator-free and more suitable for decentralized self-organizing systems. The former one is still a centralized architecture but utilizes the blockchain to ensure the authenticity of updated parameters. The latter ones leverage the decentralized blockchain consensus to aggregate the global model, which is more robust to DDoS attacks. The All-Reduce architecture can be applied in medium-scale permissioned networks (corresponding to permissioned/consortium blockchains), where workers should have permission before joining the system. The Gossip architecture can be deployed in large-scale permissionless networks (corresponding to permissionless/public blockchains), everyone can join the FL tasks and compete in the model aggregation right. Besides, they can be hybrid to obtain a trade-off between scalability and model consistency.
- *Is blockchain the best option for FL?* Blockchain is not the only solution for decentralized model aggregation, we can also use distributed Byzantine protocols to enhance the security of FL. However, blockchain with its tamper-proof feature and automatically executing smart contracts can bring unique features to FL. For example, we can use the tamper-proof feature of blockchain as a root of trust for worker rating; we can also use transparent and automatically executed smart contracts to develop a fair and open incentive mechanism and take blockchain as a reward distribution platform to solve the problem of the unfairness of centralized platform. Anyway, we need to consider the necessity of introducing blockchain before designing the BFL system.

VII. Conclusion

In conclusion, we survey a curated list of BFL-related papers from a system perspective. We deconstruct BFL systems into three layers according to the lifecycle of FL tasks. And we review existing BFL systems from these three aspects and learn some lessons, respectively. After investigating recent advances, we discuss several open challenges for designing secure, efficient, practical BFL systems. We hope this article can help readers explore potential future research directions in this field.

Acknowledgment

This work was supported in part by National Key Research and Development Program of China 2020YFB1710900, NSFC grant 62141220, 61972253, U1908212, 72061127001, 62172276, 61972254, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, Open Research Projects of Zhejiang Lab No. 2022NL0AB01. Linghe Kong is the corresponding author.

References

[1] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Security & Privacy*, vol. 19, no. 2, pp. 20–28, 2021.

[2] B. McInnis, D. Cosley, C. Nam, and G. Leshed, "Taking a hit: Designing around rejection, mistrust, risk, and workers' experiences in amazon mechanical turk," in *ACM CHI*, 2016, p. 2271–2282.

[3] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.
[4] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *IEEE BigData*, 2018, pp. 1178–1187.
[5] Z. Zhang, K. Xu, Q. Li, X. Liu, L. Li, B. Wu, and Y. Guo, "Seccl: Securing collaborative learning systems via trusted bulletin boards," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 47–53, 2020.
[6] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. A. El-Latif, "A secure federated learning framework for 5g networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 24–31, 2020.
[7] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, *FedCoin: A Peer-to-Peer Payment System for Federated Learning*. Springer International Publishing, 2020, pp. 125–138.
[8] Z. Li, H. Yu, T. Zhou, L. Luo, M. Fan, Z. Xu, and G. Sun, "Byzantine resistant secure blockchained federated learning at the edge," *IEEE Network*, vol. 35, no. 4, pp. 295–301, 2021.
[9] 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*, vol. 16, no. 6, pp. 4177–4186, 2020.
[10] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.
[11] 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*, vol. 35, no. 1, pp. 234–241, 2021.
[12] S. Zhou, H. Huang, W. Chen, P. Zhou, Z. Zheng, and S. Guo, "Pirate: A blockchain-based secure framework of distributed machine learning in 5g networks," *IEEE Network*, vol. 34, no. 6, pp. 84–91, 2020.
[13] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2021.
[14] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438–2455, 2021.
[15] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2021.

Junqin Huang (junqin.huang@sjtu.edu.cn) received the B.Eng. degree in computer science and technology from University of Electronic Science and Technology of China in 2018. He is currently persuing the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include crowdsensing, Internet of things, blockchain, mobile computing.

Linghe Kong (linghe.kong@sjtu.edu.cn) received the B.Eng. degree in automation from Xidian University in 2005, the master's degree in telecommunication from Telecom SudParis in 2007, and the Ph.D. degree in computer science from Shanghai Jiao Tong University in 2013. He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include Internet of things, 5G, blockchain, and mobile computing.

Guihai Chen (gchen@cs.sjtu.edu.cn) received the B.E. degree in computer software from Nanjing University in 1984, the M.E. degree in computer science from Southeast University in 1987, and the Ph.D. degree in computer science from the University of Hong Kong in 1997. He is a Distinguished Professor at Shanghai Jiao Tong University. His research interests include sensor networks, peer-to-peer computing, high-performance computer architecture, and combinatorics.

Qiao Xiang (qiaoxiang@xmu.edu.cn) received the bachelor's degree in information security and the bachelor's degree in economics from Nankai University in 2007, and the master's and Ph.D. degrees in computer science from Wayne State University in 2012 and 2014, respectively. He is a professor with Xiamen University. His research interests include software-defined networking, resource discovery and orchestration in collaborative data sciences, etc.

Xi Chen (xi.chen11@mcgill.ca) achieved his Ph.D. degree at School of Computer Science, McGill University. He received both M.Eng. and B.S. degrees from Department of Electronic Engineering, Shanghai Jiao Tong University. He is is currently a Senior Principle Researcher at Huawei Noah's Ark Lab, Montreal, leading a team on AI/ML applications. His research interests include AI for communications, smart IoT, WiFi sensing, etc.

Xue Liu (xueliu@cs.mcgill.ca) is a professor in the School of Computer Science and a William Dawson Scholar at McGill University. He obtained his Ph.D. in Computer Science from The University of Illinois at Urbana-Champaign and his B.S. degree in Mathematics and M.S. degree in Automatic Control both from Tsinghua University, China. He is Fellow of the Canadian Academy of Engineering (FCAE), IEEE Fellow (FIEEE), and ACM Distinghuished Member.