martFL: Enabling Utility-Driven Data Marketplace with a Robust and Verifiable Federated Learning Architecture

Qi Li
Tsinghua University & Zhongguancun Laboratory
li-q20@mails.tsinghua.edu.cn

Zhuotao Liu*
Tsinghua University & Zhongguancun Laboratory
zhuotaoliu@tsinghua.edu.cn

Qi Li
Tsinghua University & Zhongguancun Laboratory
qli01@tsinghua.edu.cn

Ke Xu
Tsinghua University & Zhongguancun Laboratory
xuke@tsinghua.edu.cn

ABSTRACT

The development of machine learning models requires a large amount of training data. Data marketplace is a critical platform to trade high-quality and private-domain data that is not publicly available on the Internet. However, as data privacy becomes increasingly important, directly exchanging raw data becomes inappropriate. Federated Learning (FL) is a distributed machine learning paradigm that exchanges data utilities (in form of local models or gradients) among multiple parties without directly sharing the original data. However, we recognize several key challenges in applying existing FL architectures to construct a data marketplace. (i) In existing FL architectures, the Data Acquirer (DA) cannot privately assess the quality of local models submitted by different Data Providers (DPs) prior to trading; (ii) The model aggregation protocols in existing FL designs cannot effectively exclude malicious DPs without "overfitting" to the DA's (possibly biased) root dataset; (iii) Prior FL designs lack a proper billing mechanism to enforce the DA to fairly allocate the reward according to contributions made by different DPs. To address above challenges, we propose martFL, the first federated learning architecture that is specifically designed to enable a secure utility-driven data marketplace. At a high level, martFL is empowered by two innovative designs: (i) a quality-aware model aggregation protocol that allows the DA to properly exclude local-quality or even poisonous local models from the aggregation, even if the DA's root dataset is biased; (ii) a verifiable data transaction protocol that enables the DA to prove, both succinctly and in zero-knowledge, that it has faithfully aggregated these local models according to the weights that the DA has committed to. This enables the DPs to unambiguously claim the rewards proportional to their weights/contributions. We implement a prototype of martFL and evaluate it extensively over various tasks. The results show that martFL can improve the model accuracy by up to 25% while saving up to 64% data acquisition cost.

*Zhuotao Liu is the corresponding author.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3623134

CCS CONCEPTS

• **Computing methodologies** → *Multi-agent systems*; • **Security and privacy** → **Privacy-preserving protocols**;

KEYWORDS

Robust Federated Learning; Data Marketplace; Verifiable Learning

ACM Reference Format:
Qi Li, Zhuotao Liu, Qi Li, and Ke Xu. 2023. martFL: Enabling Utility-Driven Data Marketplace with a Robust and Verifiable Federated Learning Architecture. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623134

1 INTRODUCTION

Artificial Intelligence (AI) continues to shape many aspects of our lives. However, the development of AI models requires a large amount of high-quality training data. However, collecting data, especially the private-domain data that is not publicly available on the Internet, is challenging. The community proposed the concept of data marketplace [30, 44, 45, 60] to address this problem. In a data marketplace (such as the International Data Spaces Association [6]), organizations can access high-quality data owned by other organizations that is specific to their needs. However, as data privacy becomes increasingly important, directly trading raw data could be inappropriate or even prohibited by laws (*e.g.,* GDPR [66], PIPL [17]). This implies a fundamental paradigm shift from trading raw data to only trading data utilities without raw data exchange.

Federated Learning (FL) [58] is a machine learning paradigm that enables multiple parties to train a global model on their own data without sharing the data with each other. This is achieved by having each party train a local model on their own data and then sending the updates to a central server. The central server then aggregates the updates from all of the parties to create a global model. This makes FL a promising paradigm for a utility-driven marketplace because organizations can buy and sell data without having to share the underlying data. Viewing FL as a primitive, we could construct a strawman data marketplace as shown in Figure 1(a). In this diagram, the aggregation server in FL serves as the *data acquirer* (DA) that initiates the FL task. The FL clients serve as the *data providers* (DPs) to participate in the FL task by providing local model updates trained on their own data. The DA evaluates the local models submitted by different DPs to purchase high-quality

local models. Eventually, the DA aggregates local models to update the global model, based on which it may initiate another iteration.

However, we recognize three major challenges in applying the vanilla FL to construct a secure data marketplace. First, the model aggregation protocol in vanilla FL does not allow the DA to evaluate the data quality of local model updates before obtaining the updates from DPs. This raises a dilemma in data trading: the DPs are unwilling to give away their local updates before receiving rewards, while the DA prefers to evaluate the updates first before purchasing them.

Second, the model aggregation protocol of the vanilla FL is subject to various attacks, such as [9, 31, 52, 76]. Prior art on mitigating these issues can be roughly categorized into client-driven approaches and server-driven approaches. The client-driven approaches [12, 71, 72] improve aggregation robustness by smoothing the local update updates based on their statistics (*e.g.,* median or average). The server-driven approaches [18, 56] instead rely on the DA to lead the aggregation process. They assume that the DA possesses a high-quality *root dataset* based on which it can calibrate the local model updates submitted by the DPs. As a result, prior works exhibit a fundamental tradeoff between inclusiveness and robustness: the client-driven approaches can potentially include more local model updates for training, yet they stake robustness on the "honest majority" (which might be incorrect in the data trading scenario), while the server-driven approaches are more resilient against malicious DPs, yet they sacrifice inclusiveness by "overfitting" to the DA's existing dataset (which could be biased).

Third, existing model aggregation protocols lack the required *verifiability* to enable fair billing. Specifically, in FL, the local models receiving higher aggregation weights have more impacts on the final model. Thus, the aggregation weights essentially quantify the values (or utilities) provided by the local model updates. Prior art (*e.g.,* Omnilytics [51], FPPDL [57]) tries to achieve fair billing by directly executing *the entire model aggregation process* on blockchains, which significantly limits the design space of the aggregation algorithms (see analysis in § 2.2). Thus, the third challenge in designing a fair utility-driven data marketplace is to ensure that the DA faithfully distributes rewards among the DPs according to their actual aggregation weights. This also ensures that the DA only pays for its desired model updates, rather than blindly purchasing arbitrary updates, which greatly reduces model acquisition cost (see evaluation results in § 6.2).

To address these challenges, we present martFL, a novel FL architecture that enables robust and verifiable local model aggregation in a utility-driven data marketplace. martFL is powered by two innovative designs. First, martFL designs a two-phased protocol that first privately evaluates all local model updates submitted by DPs based on a baseline to remove outliers (*i.e.,* local-quality updates) and then dynamically adjusts the evaluation baseline to incorporate the high-quality updates. Therefore, our quality-aware model aggregation protocol eliminates the fundamental tradeoff between inclusiveness and robustness, by indiscriminately evaluating the complete set of DPs and meanwhile avoiding overfitting to the (possibly biased) root dataset owned by DA.

Second, martFL designs a novel verifiable data transaction protocol that enables the DA and the selected DPs to securely exchange the reward and model updates. Our verifiable transaction protocol centers around a proving scheme that allows the DA to prove, both succinctly and in zero-knowledge, that it faithfully aggregates the model using the committed aggregation weights. Based on the publicly verifiable proof, the DPs can unambiguously claim the reward corresponding to their weights. Crucially, martFL achieves the fair trading without relying on any online trusted third party to regulate the trading process.

Contributions. The main contribution of this paper is the design, implementation and evaluation of martFL, the first FL architecture that simultaneously offers robustness and verifiability to enable a secure utility-driven data marketplace. We implement a prototype of martFL in approximately 3750 lines of code and extensively evaluate its accuracy and robustness using two image classification datasets and two text classification datasets. The results show that compared to existing server-driven methods, martFL can improve accuracy by up to 25% even when the DA has a biased root dataset, while saving up to 64% data acquisition cost. In addition, martFL can resist various untargeted attacks, targeted attacks, and Sybil attacks, and achieves the highest accuracy and the lowest attack success rate compared to both server-driven and client-driven methods. We also report the system-level overhead of martFL to demonstrate its feasibility in practice.

2 BACKGROUND AND MOTIVATION

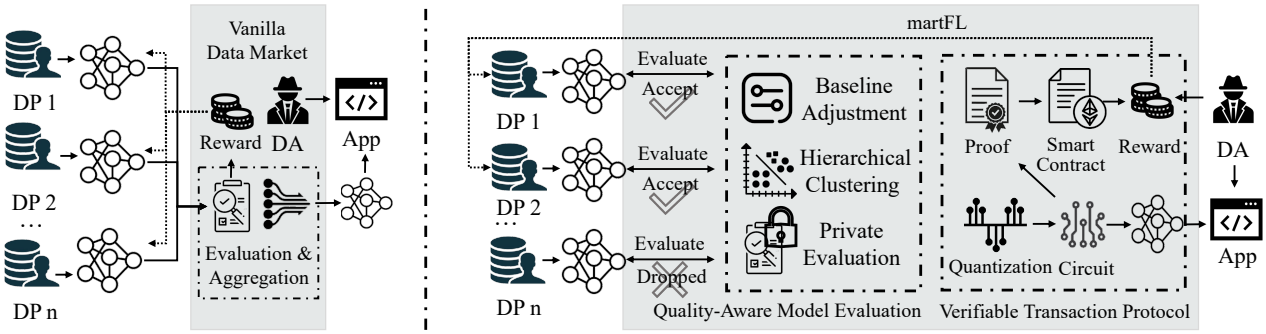
2.1 Data Marketplace

The traditional circulation of data trading mainly relies on data trading platforms (such as International Data Spaces [6], BDEX [3], Quandl [7] and GE Predix [5]) that are endorsed by government or industry leaders. The research community explored API-based marketplace designs that allow the data acquirers to collect data stream online [45]. Due to the rising importance of data privacy, direct trading of raw data, particularly data associated with personal information [17, 66], is subject to significant regulatory burdens in practice. Therefore, it is essential to explore data marketplaces that do not require direct exchange of raw data.

2.2 Federated Learning and Its Robustness

In designing an AI-specific marketplace, Federated Learning (FL) [58] is a promising learning paradigm since it enables collaborative training without directly sharing the raw data. A data marketplace built upon the vanilla FL architecture has three phases: (i) global model distribution: the central server (serving as the data acquirer DA) initializes a global model and distributes it to the clients (serving as the data providers DPs); (ii) local model training: the DPs use their local data to train the model and then upload the resulting models (referred to as local models) to the DA; and (iii) model aggregation: the DA aggregates these local models to obtain a new global model. This process repeats for multiple epochs until the DA obtains a sufficiently accurate global model.

However, the above vanilla FL-driven data marketplace faces several critical challenges. First, FL is known to be vulnerable to various attacks, such as untargeted attack [11, 27] (*e.g.,* the Byzantine clients disrupt the training process by rescaling the sizes of local gradients or randomizing the directions of local gradients), targeted attack [9] (*e.g.,* the Byzantine clients mislead the global model to



(a) Vanilla data market (b) martFL

Figure 1: The architecture comparison between the vanilla FL and martFL.

specifically misclassify certain classes), and Sybil Attack [31]. The community has therefore proposed various robustness FL designs that can be roughly divided into two categories. The client-driven approaches [11, 12, 31, 71, 72] try to exclude malicious local model updates by learning representative statistics from all local models; and the server-driven designs [18, 19, 27, 56] instead assume that the server owns a trusted root dataset based on which it can calibrate these local models. These approaches suffer from a fundamental tradeoff between inclusiveness and robustness, resulting in non-trivial performance degradation (see § 4.1).

In addition to the robustness concern, existing FL architectures lack several key features that are essential for data trading. On the one hand, the data acquirer (DA) cannot assess quality of the local models submitted by different DPs prior to trading; on the other hand, the DPs are not assured of receiving adequate compensation after submitting their models. Several recent approaches (*e.g.,* Omnilytics [51], FPPDL [57]) try to achieve trading-oriented FL designs by simply executing the *entire model aggregation process* on blockchains, either via general-purpose smart contracts or leveraging specialized block structures. These approaches, however, are fundamentally limited because they force the DA to make the local model assessment protocol publicly executable on blockchains, preventing the DA from using proprietary and complex/advanced algorithms. As a result, the aggregation algorithm in FPPDL [57] is unable to handle malicious DPs; and Omnilytics [51] only supports four DPs using the simple Secure-Aggregation algorithm [13] with the multi-Krum [12] algorithm to remove outliers, while incurring significant gas cost (at least 1000 times more than martFL, as shown in § 6.2.5). martFL is fundamentally different from these blockchain-based FL approaches because martFL relies on smart contract to verify the correctness of *the offline model assessment and aggregation* performed by the DA. This enables the DA to design proprietary and advanced local model evaluation protocols to handle various FL attacks. Additionally, martFL designs a verifiable transaction protocol to ensure the DA cannot cheat about the reward allocation, even though the DA uses proprietary model aggregation protocols that are not known to the DPs.

2.3 Zero-Knowledge Proofs

To ensure fair trading, martFL requires the DA to publicly prove that it has faithfully aggregated local models using the aggregation weights that were committed to before receiving the plaintext local models from the DPs. This proving process can be formulated as an argument of knowledge for the aggregation protocol, without disclosing these local models to the public. The recent progress in zero-knowledge proof technology, especially the development of zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK) [15, 37, 42, 67] where the prover only needs present one message (proof) instead of interacting with the verifier [34], has demonstrated the potential to achieve this goal. Yet, simply applying existing zk-SNARK constructions [16, 22, 37] to prove the end-to-end training process in FL is challenging. This is because (i) the detailed local model evaluation algorithm can be complex and even contains computations over homomorphically encrypted values (see § 4.2); and (ii) the model sizes are large, for instance, with millions of floating-point parameters or even more. Both of these issues would result in significantly large proof circuits, which are impractical to implement.

2.4 Motivation

To address above challenges, we propose martFL, a secure and verifiable FL architecture specifically designed for utility-driven data marketplaces. martFL advances state-of-the-art in both secure local model aggregation and verifiable data trading. In particular, martFL designs a novel quality-aware model evaluation protocol that can indiscriminately and privately assess all the local models submitted by the DPs based on a dynamically adjusted baseline. As a result, it can accurately remove malicious local models while avoiding overfitting to the root dataset owned by the DA, eliminating the tradeoff between inclusiveness and robustness exhibited in prior art. Further, martFL proposes an efficient verifiable transaction protocol that enables fair data trading *without the need to prove the entire FL training process.* The key novelty of our approach is that our proving scheme focuses on only proving the critical computation that is necessary and sufficient to ensure fair billing. This results in the proving overhead being independent of both the local model evaluation algorithm and the model size. Given this proof, the DPs can unambiguously claim corresponding reward over a smart contract. To the best of our knowledge, this is the first verifiable scheme designed specifically for proving the correctness of model aggregation in FL, without directly placing the entire model aggregation protocol on blockchains.

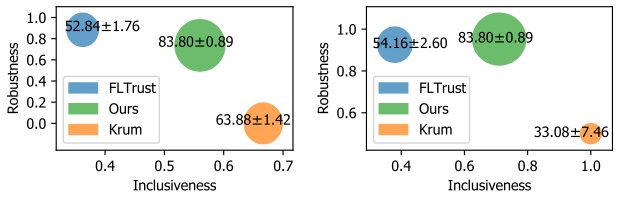
2.5 Assumptions and Threat Model

We consider Byzantine DPs that may submit arbitrary local models. They may launch these aforementioned attacks to disrupt the training process, or try to earn rewards without actual contributions to training (*e.g.,* the free-rider attack [52]). We consider that the DA is semi-honest, *i.e.,* the DA is protocol-compliant, but motivated to manipulate the reward distribution so as to minimize the cost of collecting data. We assume that the DA possesses a root dataset. Many well-established robust FL approaches (*e.g.,* [18, 27]) assumed that DA has a *reliable and unbiased* root or validation dataset to handle malicious DPs. In contrast to these approaches, the root dataset assumed in martFL can be *both of poor quality and of limited volume*. For instance, it may contain only half of the labels (*i.e.,* the DA's root dataset exhibits biased distributions), or it may be approximately 1% of the data held by all DPs (see evaluation results in § 6.2.1). Therefore, the assumption made about the root dataset in martFL is significantly less restrictive than that made by existing robust FL approaches. This makes martFL suitable for the data trading scenario, in which the DA, without necessarily possessing a good root dataset, can collect high-quality and high-volume local models from a diverse set of DPs.

We assume that the cryptographic primitives and the consensus protocol of the blockchain system used to host the data transaction smart contract in martFL are secure so that the blockchain can have the concept of transaction finality and contract publicity. On Nakamoto consensus based blockchains, finality is achieved by assuming that the probability of blockchain reorganizations drops exponentially as new blocks are appended (*i.e.,* the common-prefix property) [32]. On Byzantine tolerance based blockchains, finality is guaranteed by signatures from a quorum of permissioned voting nodes. We assume that the blockchain has a public ledger that allows external parties to examine the public state of its deployed smart contracts. We assume that the zk-SNARK protocol [37] used in our verifiable transaction protocol is sound.

3 MARTFL OVERVIEW

Architecturally, martFL is designed around four components (as shown in Figure 1(b)). (i) A data acquirer (DA) relies on martFL to collect training data for a FL training task from a utility-driven marketplace like martFL. Each training epoch is associated with a reward that the DA will pay after the data trading is closed. (ii) Data providers (DPs) participate in FL training by contributing their local model updates. martFL itself has two building blocks. (iii) A Quality-aware Model Evaluation Protocol that enables the DA to confidentially pre-evaluate the quality of the local models from different DPs. The DA can keep the detailed aggregation algorithm (*e.g.,* how to remove poisonous local models) confidential, making it difficult for the malicious DPs to manipulate the training process (see analysis in § 6.3.2). (iv) Afterwards, they apply the Verifiable Transaction Protocol to achieve fair data trading. The DA first commits the *aggregation weights*, obtained by the model evaluation protocol, on the trading smart contract. Upon commitment, the DPs can safely submit their plaintext local models offline to the DA. The DA is expected to generate a publicly verifiable proof (without disclosing its model evaluation method and the received local models) to demonstrate that it has faithfully aggregated these local models using the committed weights. Given the proof, the DPs can unambiguously claim the reward (proportional to their aggregation weights) deposited by the DA on the smart contract. Violations against the transaction protocol (*e.g.,* the proof verification fails) results in automatic penalties coded in the smart contract.



(a) Targeted Attack (b) Untargeted Attack

Figure 2: The tradeoff between robustness and inclusiveness in prior robust FL approaches.

4 QUALITY-AWARE MODEL EVALUATIONS

4.1 Key Observations

We first discuss the key observations about the tradeoff between the inclusiveness and robustness in the prior client-driven and server-driven secure FL aggregation protocols, which motivates our model aggregation design. We consider a common data trading scenario where the DA has an unevenly distributed root dataset prior to trading, and the data qualities for different DPs vary and some DPs are malicious. Specifically, using the TREC dataset [50] as an example, suppose that (i) the root dataset of the DA is dominated by half of the class labels; (ii) the DPs are heterogeneous, where 30% of them have high-quality data (evenly distributed across all types of labels), 30% of them own biased dataset, and 40% of them are malicious; (iii) the malicious DPs may launch the backdoor attack [9] (a type of the targeted attack) or the sign-randomizing attack (a type of untargeted attack). We evaluate two representative prior art using this setting: a server-driven design FLTrust [18] and a client-driven design Krum [12].

We report three metrics in Figure 2. Robustness represents the ability to exclude poisoned local models, quantified by the percentage of malicious DPs whose local models are not selected for aggregation. Inclusiveness represents the ability to identify benign DPs, quantified by the percentage of benign DPs whose local models are selected for aggregation. Accuracy represents the final model performance on the testset. We observe a clear tradeoff between inclusiveness and robustness in prior art, where the server-driven approach has higher robustness while only selecting DPs similar to the (biased) root dataset (sacrificing inclusiveness), and the client-driven design behaves the opposite. Instead, our design strikes a good balance between robustness and inclusiveness, thus yielding significant accuracy gain over prior art. In § 6.3.1, we further investigate this tradeoff using a series of different parameters.

4.2 Model Aggregation Protocol

The architecture of our local model evaluation protocol is presented in Algorithm 1. The DA first prepares a baseline model using its own root dataset (could be biased). This model will be used as a reference for scoring other local models submitted by DPs in each training

Algorithm 1: Quality-Aware Model Aggregation Protocol

1 **Inputs:** The scores of local models in the t-th training epoch $\mathcal{S}^t = \{s_1^t, s_2^t, \ldots, s_n^t\}$; the DP selected as the baseline in the t-th epoch p^t; a control flag α for baseline adjustment; the ratio of randomly selected baseline candidates β; the threshold T used in hierarchical clustering; the root dataset D_0; the maximum number of clusters G.
2 **Outputs:** The aggregation weights obtained for the t-th epoch and the DP selected as the baseline for the $(t+1)$-th epoch p^{t+1}.
3 ---
4 **Function Main** $(\mathcal{S}^t, p^t, \alpha, \beta, T, D_0, G)$ **:**
5 *// Set $\mathcal{P}^t$ stores the DPs selected for aggregation; Set $\mathcal{K}^t$ stores their weights.*
6 $\mathcal{P}^t, \mathcal{K}^t \leftarrow$ OutlierRemoval $(\mathcal{S}^t, p^t, \beta, T, G)$
7 *// $\mathcal{M}^t$ are the plaintext models that the DA commits to purchase.*
8 $\mathcal{M}^t \leftarrow$ ModelTrading$(\mathcal{P}^t)$
9 **if** α = **true then** $p^{t+1} \leftarrow$ BaselineAdjustment$(\mathcal{M}^t, D_0)$
10 **else** $p^{t+1} \leftarrow 0$
11 ---
12 **Function OutlierRemoval**$(\mathcal{S}^t, p^t, \beta, T, G)$ **:**
13 $\mathcal{U} \leftarrow \{1, 2, \ldots, n\}$, $\mathcal{P}_1 \leftarrow \emptyset$, $\mathcal{P}_2 \leftarrow \emptyset$, $\mathcal{K} \leftarrow \{1.0, \ldots, 1.0\}$
14 *// Determine the number of clusters $\hat{g}$ by Gap statistics.*
15 **for** $g \leftarrow 1, 2, \ldots, G$ **do**
16 $\quad \hat{g} \leftarrow$ the minimum g such that $\text{Gap}(g) - \text{Gap}(g+1) + \sigma_{g+1} \geq 0$
17 $d \leftarrow \text{Max}(\mathcal{S}^t) - \text{Min}(\mathcal{S}^t)$
18 **if** $\hat{g} = 1$ **and** $d > T$ **then** $\hat{g} \leftarrow 2$
19 **else** $\mathcal{P}_1 \leftarrow \mathcal{U}$ *// Single-cluster gathered distribution.*
20 *// K-Means returns the clusters and centroids of the scores.*
21 $\mathcal{N}_1, C_1 \leftarrow$ K-Means$(\mathcal{S}^t, \hat{g})$
22 $C_{best} = \text{Max}(C_1)$ *// Centroid of the highest-score cluster.*
23 **if** $\hat{g} > 2$ **then** $\mathcal{N}_2, C_2 \leftarrow$ K-Means$(\mathcal{S}^t, 2)$ *// Re-clustering.*
24 **else** $\mathcal{N}_2 \leftarrow \mathcal{N}_1$
25 **for** $i \leftarrow 1, 2, \ldots, n$ **do**
26 $\quad$ **if** $\hat{g} = 1$ **then** break
27 $\quad$ **if** $i = p^t$ **or** $\mathcal{N}_1[i] = 0$ **or** $\mathcal{N}_2[i] = 0$ **then**
28 $\quad\quad \mathcal{K}[i] \leftarrow 0.0$ *// Low-quality model.*
29 $\quad$ **else if** $\mathcal{N}_1[i] = \hat{g} - 1$ **and** $(\mathcal{N}_2[i] \neq 0)$ **then**
30 $\quad\quad \mathcal{K}[i] \leftarrow 1.0$, $\mathcal{P}_1$.add(i) *// High-quality model.*
31 **else**
32 $\quad \mathcal{K}[i] \leftarrow 1.0 - \frac{\text{Abs}(\mathcal{S}[i] - C_{best})}{\text{Max}(\text{Abs}([s_i^t - C_{best} \textbf{ for } s_i^t \textbf{ in } \mathcal{S}]))}$
33 $\quad \mathcal{P}_2$.add(i) *// Qualified but weighted model.*
34 **if** $\mathcal{P}_2 = \emptyset$ **and** $\text{Len}(\mathcal{P}_1) < 0.5 \times n$ **then**
35 $\quad \mathcal{P}_2 \leftarrow \text{RandomSample}(\mathcal{U} - \mathcal{P}_1, \beta)$
36 **return** $\mathcal{P}_1 \bigcup \mathcal{P}_2$, $\frac{\mathcal{K}}{\text{Sum}(\mathcal{K})}$
37 ---
38 **Function BaselineAdjustment**$(\mathcal{M}^t, D_0)$ **:**
39 kp_{max} = -inf, $p^{t+1} = 0$
40 **for** i, m **in Enumerate**$(\mathcal{M}^t)$ **do**
41 $\quad kp \leftarrow \text{Kappa}(m, D_0)$
42 $\quad$ **if** $kp > kp_{max}$ **then** $kp_{max} \leftarrow kp, p^{t+1} \leftarrow i$
43 **return** p^{t+1}

epoch. Afterwards, the DA clusters the DPs according to their scores and removes the outliers (*i.e.,* low-quality local models) for this epoch. Finally, the DA and selected DPs finalize the data trading using our verifiable transaction protocol detailed in § 5, which guarantees that the DA distributes rewards to the DPs according to their model quality. Before starting the next training epoch, the DA *dynamically adjusts* the baseline to incorporate the high-quality data collected in the prior epoch, which is the key to address the possibly biased root dataset. Throughout the training process, we apply Homomorphic Encryption (HE) to ensure that the DA cannot obtain plaintext local models before committing to purchase them.

4.2.1 *Hierarchical Clustering for Outlier Removal*

We score the local models submitted by DPs using cosine similarities (similar to FLTrust [18]). Suppose that W_g^t is the global model at round t, $W_g^{t'}$ is the baseline model (in the first epoch, it trained from W_g^t by the DA with its root dataset D_0), and $u_g^t = \text{Flatten}(W_g^{t'} - W_g^t)$ is therefore the self-update computed by the DA. Suppose that W_i^t is the model obtained by the i-th DP after it trains W_g^t on its local dataset D_i, and $u_i^t = \text{Flatten}(W_i^t - W_g^t)$ is the update computed by the i-th DP. Then, the score of u_i^t is calculated as follows.

$$s_i^t = \text{Cosine}(u_g^t, u_i^t) = \frac{u_g^t \cdot u_i^t}{||u_g^t|| \cdot ||u_i^t||} \tag{1}$$

The DA selects the desired updates according to their scores. Unlike the FLTrust [18] that simply clips the scores via ReLU, our design avoids simply referencing the DA's root dataset by analyzing the cluster distribution of all scores. Specifically, we propose a hierarchical clustering algorithm to select the desired updates. We first apply the Gap-Statistics algorithm [65] to determine the optimal number of clusters $\hat{g}$ [1]. Afterwards, we obtain our first-layer clustering by applying the K-Means algorithm [55] with $\hat{g}$. This may produce three types of distributions, as shown in Figure 3.

- Single-cluster gathered distribution: the model scores are concentrated, and the range of scores is less than a predefined threshold T. This often indicates models submitted by all DPs have comparable quality. In this case, we include all updates to the high-quality model set $\mathcal{P}_1$ (line 19 of Algorithm 1).
- Single-cluster scattered distribution: the model scores are scattered over a large range. This could be a sign of attack, where the malicious DPs intentionally submit arbitrary model updates. As a result, we perform the second layer of clustering (via K-Means clustering with $\hat{g}$=2) to divide the scores into a high-quality cluster and a low-quality cluster. The updates in the high-quality cluster are selected for aggregation in this round (line 30 in Algorithm 1) .
- Multi-cluster distribution: the update scores form multiple clusters. This could be caused by highly-heterogeneous DPs where some of them possess good dataset, some of them possess biased dataset and some of them are malicious. The algorithm performs the second clustering by separating these first-stage clusters into two categories. The low-quality category is discarded. Within the high-quality category, the updates in the highest-score cluster are added to the set $\mathcal{P}_1$. The local models in the remaining clusters of the high-quality category considered to be qualified

[1]The elbow coefficient [64] and silhouette [63] are other possible methods to calculate $\hat{g}$. However, the elbow coefficient algorithm requires manual judgment to determine the position of the elbow, and the silhouette algorithm can only be used with two or more clusters.

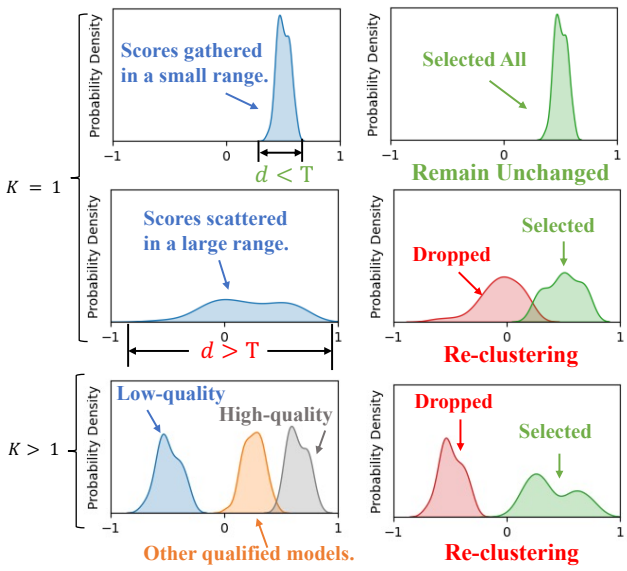


Figure 3: Three different cases of distribution of scores.

(line 32 of Algorithm 1), but weighted based on their distances to the centroid of the highest-score cluster.

Eventually, the DPs in $\mathcal{P}_1$ and a small subset of DPs (*e.g.*, 5%-10%) randomly selected from $\mathcal{P}_2$ are selected for aggregation. The DA first commits to purchase local models from these DPs. Afterwards, it is safe for the selected DPs to hand over the plaintext local models to the DA (see the detailed transaction protocol in § 5).

4.2.2 *Dynamic Baseline Adjustment*

To avoid overfitting to the DA's root dataset, martFL enables the DA to dynamically adjust the baseline for outlier removal. Specifically, for each local model $m \in \mathcal{M}^t$, the DA evaluates m on its root dataset and computes the Kappa coefficient [23]. The DA then selects the DPs with high Kappa coefficients as *preferred DPs*. For simplicity, Algorithm 1 only selects the DP with the highest-Kappa-coefficient as the single preferred DP (line 42). In the next epoch, the DA trades the local models in advance with these preferred DPs and aggregates them as the new baseline. The DA should not disclose these preferred DPs until they have committed their local models.

4.3 The Integrated Training Process

We have described the local model evaluation protocol in martFL. In a utility-driven data marketplace, it is critical to ensure that the DA cannot obtain the plaintext model updates before committing to purchase them. Towards this end, we apply the CKKS Homomorphic Encryption [21] to allow DA to privately assess the local models submitted by the DPs.

Supposed that in the t-th epoch, the DA obtains the baseline update u_g^t. Instead of directly sharing u_g^t with the DPs, the DA homomorphically encrypts it by the public key k as $c_g^t = \mathsf{Enc}(k, \frac{u_g^t}{||u_g^t||})$. Once a DP receives c_g^t, it multiply its local update u_i^t with c_g^t as $c_i^t = \frac{u_i^t}{||u_i^t||} \cdot c_g^t$, and returns the result back to the DA. Eventually, the DA receives the encrypted cosine c_i^t, and decrypts it to obtain the score for the update u_i^t. Afterwards, the DA can perform local model evaluations as described in § 4.2.

A critical step in adopting CKKS is to safeguard against the DPs from using different models in model evaluation and subsequent model transactions, *i.e.,* preventing the DPs from intentionally submitting different model updates after being selected by the DA. To this end, we require the DPs to commit their model updates before model evaluations. These committed updates are then used to ensure the correctness of subsequent model transactions, as we will further discuss below.

5 VERIFIABLE TRANSACTION PROTOCOL

Our verifiable transaction protocol has two phases: (i) a zero-knowledge proving system that allows the DA to prove that it has faithfully aggregated the global model based on claimed weights, without disclosing the local models submitted by the DPs; and (ii) a payment protocol based on smart contract to allow the DA and DPs to exchange rewards and plaintext local models.

5.1 Proving Scheme for Model Aggregation

5.1.1 *Overview*

The DA should prove that it faithfully aggregates the global model. Although there are many zero-knowledge proof (ZKP) constructions [2, 4, 26], it is challenging to simply adopt these designs to achieve verifiable aggregation in martFL. Specifically, the model evaluation algorithm (Algorithm 1) used by martFL is complex, especially considering the homomorphic computations involved. This complexity makes it difficult to generate and implement the arithmetic circuit to represent the algorithm. To address this challenge, martFL does not prove the end-to-end training process. Instead, it only proves the local model summation computation, which aggregates the local models using the aggregation weights returned by the model evaluation algorithm. This design drastically reduces the proving complexity without affecting the fairness of billing, because reward allocations are completely driven by the aggregation weights. In addition, we also design verifiable sampling method such that the DA only needs to prove a fix number of scalars/parameters regardless of the model size (*i.e.,* the number of model parameters).

Setup. Denote the local model summation as $\mathcal{A}$, which represents the following calculation $W_g^t = W_g^{t-1} + K^t U^t$, where W_g^{t-1} (W_g^t) is the global model in the previous (current) epoch. For each $k_i^t \in \mathcal{K}^t$, $K^t = [k_1^t, k_2^t, \ldots, k_n^t]$ are the aggregation weights claimed by the DA, and $U^t = [u_1^t, u_2^t, \ldots, u_n^t]$ are the local models submitted by the DPs. The public input of our zero-knowledge proving scheme is $\mathcal{X}^t = \{W_g^t, W_g^{t-1}, K^t\}$, and the private witness is $\mathcal{W}^t = \{U^t\}$. Concretely, our proving scheme has the following algorithms.

- $\mathbb{C} \leftarrow$ Compile($\mathcal{A}$): In the compiling step, the prover (*i.e.,* the DA) quantizes the floating-point public input $\mathcal{X}^t$ and private witness $\mathcal{W}^t$ to $\mathbb{X}^t = \{\mathbb{W}_g^t, \mathbb{W}_g^{t-1}, \mathbb{K}^t\}$ and $\mathbb{W}^t = \{\mathbb{U}^t\}$ in finite field, respectively. In addition, it quantizes the aggregation algorithm $\mathcal{A}$ and compiles it to a circuit $\mathbb{C}$.
- $(pk, vk) \leftarrow$Setup($1^\lambda, \mathbb{C}$): Given a security parameter λ and the circuit $\mathbb{C}$, a trusted third party randomly generates a proving key pk and a verification key vk. The proving key pk is given to the

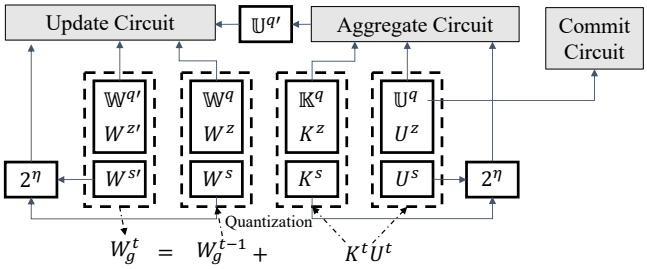


Figure 4: The circuit design for the proving scheme in martFL.

DA and the verification key vk is given to the DPs. We consider a proving scheme that requires trusted setup in this paper, and leave exploration of trust-free schemes in future work.

- $(cm^t, \mathbb{W}_g^t, \pi^t) \leftarrow \text{Prove}(\mathbb{X}^t, \mathbb{W}^t, r^t, pk, \mathbb{C})$: Given a random opening r^t, the prover first commits the private witness $\mathbb{W}^t$ as $cm^t = \text{Commit}(\mathbb{U}^t, r^t)$. Then it calculates the quantized global model $\mathbb{W}_g^t$ and generates a proof π^t. Afterwards, the prover publishes cm^t, π^t, and $\mathbb{W}_g^t$ to the DPs.
- $\{1, 0\} \leftarrow \text{Verify}(\mathbb{X}^t, vk, \pi^t, cm^t)$: The public verifiers (*e.g.,* DPs) can verify the computation in $\mathbb{C}$ using the verification key vk, public input $\mathbb{X}^t$, the commitment cm^t, and the proof π^t. If the DA faithfully aggregates the global model, the verifier will accept the proof; otherwise, the verifier will reject it.

5.1.2 Circuit Design

Quantization The circuit $\mathbb{C}$ is designed based on the quantized version of algorithm $\mathcal{A}$. Quantization maps a floating point value $x \in [a, b]$ to an unsigned integer $x_q \in [a^q, b^q]$ and de-quantization is the reversed process. As defined in the partial quantization [38], the quantization and de-quantization are represented as $q = \lfloor \frac{x}{s} \rfloor + z$ and $x = s(q - z)$, respectively, where s is a floating-point scaler, q is the quantized integer for x, and z is the zero point (*i.e.,* the value of a floating-point zero when mapped to the integer field).

We observe in our experiments that the above quantization design may result in overflow. Specifically, the subtractions on unsigned integers may result in overflow due to accuracy loss in quantization. Consequently, the de-quantization produces a very inaccurate dequantized model for the next training epoch. To avoid overflow, we extend the range of floating-point numbers by a small ϵ, *i.e.,* $x \in [a - \epsilon, b + \epsilon]$. Afterwards, we derive s and z, by solving the following linear Equation (2).

$$a - \epsilon = s(a^q + z); \quad b + \epsilon = s(b^q + z) \tag{2}$$

Commitment Circuit. The first part of $\mathbb{C}$ is to commit the private parameters $\mathbb{U}^t$ with the opening random r^t such that $\mathbb{U}^t$ are not disclosed to the public verifiers, *i.e.,* $cm^t = \text{Commit}(\mathbb{U}^t, r^t)$. POSEIDON [36] is an optimized commitment algorithm. Yet, simply applying POSEIDON to commit all model parameters would require in a large number of constraints. In § 5.1.3, we design a verifiable sampling mechanism to avoid committing and verifying all parameters.

Aggregation Circuit Design. The second part of $\mathbb{C}$ is the aggregation circuit that computes $U^{t'} = K^t U^t$ in the quantized form, where $K^t \in \mathbb{R}^{1\times n}$, $U^t \in \mathbb{R}^{n\times m}$, $U^{t'} \in \mathbb{R}^{1\times m}$, n is the number of DPs, and m is the number of parameters in the model. To be ZKP-friendly, we minimize the use of negative numbers and division in the calculation, while ensuring that all operations are performed in the field $\mathbb{F}_q$. First, in Equation (3), we apply the de-quantization equation.

$$U^{s'}(\mathbb{U}_{i,j}^{q'} - U^{z'}) = \sum_{k=1}^{n} K^s(\mathbb{K}_{i,k}^{q} - K^z)U^s(\mathbb{U}_{k,j}^{q} - U^z) \tag{3}$$

where $\mathbb{K}^q$, $\mathbb{U}^q$ and $\mathbb{U}^{q'}$ are the quantization matrix of K^t, U^t, and $U^{t'}$, respectively; K^s, U^s and $U^{s'}$ are the scaler of K^t, U^t, and $U^{t'}$, respectively; K^z, U^z and $U^{z'}$ are the zero points of K^t, U^t, and $U^{t'}$, respectively. In the context, $\mathbb{K}^t = \{\mathbb{K}^q, K^s, K^z\}$, etc. We use a big integer 2^η (η should be 22 or even larger) to replace the floating-point scale with unsigned integers and enable the full quantization computation. Also, we rearrange the calculation order in Equation (4) to eliminate negative numbers in calculation. The remainder $\mathbb{R}^a$ is to ensure correctness after division, as shown in [29].

$$\begin{aligned} &2^\eta \mathbb{U}_{i,j}^{q'} = \mathbb{R}_{i,j}^{a} + 2^\eta U^{z'} + \left\lfloor 2^\eta \frac{K^s U^s}{U^{s'}} \left(M_1 + M_4 - M_2 - M_3 \right) \right\rfloor \\ &\text{s.t. } M_1 = \sum_{k=1}^{n} \mathbb{K}_{i,k}^{q} \mathbb{U}_{k,j}^{q}, M_2 = U^z \sum_{k=1}^{n} \mathbb{K}_{i,k}^{q}, \\ &M_3 = K^z \sum_{k=1}^{n} \mathbb{U}_{k,j}^{q}, M_4 = nK^z U^z \end{aligned} \tag{4}$$

Update Circuit Design. The third part of $\mathbb{C}$ is the update circuit. We use Equation (5) to present the de-quantized update equation $W_g^t = W_g^{t-1} + U^{t'}$, with $W_g^t \in \mathbb{R}^{1\times m}$, $W_g^{t-1} \in \mathbb{R}^{1\times m}$, $U^{t'} \in \mathbb{R}^{1\times m}$.

$$W^{s'}(\mathbb{W}_{i,j}^{q'} - W^{z'}) = W^s(\mathbb{W}_{i,j}^{q} - W^z) + U^{s'}(\mathbb{U}_{i,j}^{q'} - U^{z'}), \tag{5}$$

where $\mathbb{W}^{q'}$, $\mathbb{W}^q$ and $\mathbb{U}^{q'}$ are the quantization matrices of W_g^t, W_g^{t-1} and $U^{t'}$, respectively; $W^{s'}$, W^s and $U^{s'}$ are the scaler of W_g^t, W_g^{t-1} and $U^{t'}$, respectively. $W^{z'}$, W^z and $U^{z'}$ are the zero points of W_g^t, W_g^{t-1} and $U^{t'}$, respectively.

Similarly, we rearrange the above equation to Equation (6) to eliminate negative numbers. And the remainder $\mathbb{R}^u$ to ensure correctness after division.

$$\begin{aligned} &2^\eta \mathbb{W}_{i,j}^{q'} = \mathbb{R}_{i,j}^{u} + 2^\eta W^{z'} + \left\lfloor 2^\eta \left(N_1 + N_3 - N_2 - N_4 \right) \right\rfloor \\ &\text{s.t. } M_1 = \frac{W^s}{W^{s'}} \mathbb{W}_{i,j}^{q}, N_2 = \frac{W^s}{W^{s'}} W^z, \\ &N_3 = \frac{U^{s'}}{W^{s'}} \mathbb{U}_{i,j}^{q'}, N_4 = \frac{U^{s'}}{W^{s'}} U^{z'} \end{aligned} \tag{6}$$

In summary, the complete circuit $\mathbb{C}$ is plotted in Figure 4.

5.1.3 Verifiable Sampling

Given the concatenated local models $U^t \in \mathbb{R}^{n\times m}$ (where n is the number of DPs, and m is the number of parameters in the model), the number of constraints required in the commitment circuit, the aggregation circuit, and the update circuit is $O(H{\cdot}n{\cdot}m)$, $O(n{\cdot}m)$ and $O(m)$, respectively, where H represents the required constraints in the hash function used in commitment circuit. Considering that

$n \ll m$ and H is fixed once the commitment hash function is selected, we explore to reduce the number of parameters required for proof generation. Specifically, we randomly select c out of m parameters as the verification objects. Suppose that the sampling is provable random (*i.e.,* not controlled by the DA), as long as the DA can provide the correct proof for the sampled parameters, then with high probability, the DA has calculated all parameters correctly. Thus, the proof complexity becomes independent on m.

Conceptually, the provable random sampling is similar to randomness beacon [62]. Both verifiable random function (VRF) [35, 59] and verifiable delay function (VDF) [14] can be used as a primitive to construct the verifiable random sampling. We sketch a construction below. In each training epoch, each DP publishes a cryptographic nonce to a public bulletin board (*e.g.,* a public blockchain). The DA is required to use $\mathcal{H}(s_1, s_2, ..., s_n)$ as the seed s^t_{vdf} to a VDF to select the parameter indices $R^t_{\text{vdf}} = \{r^t_1, r^t_2, \ldots, r^t_c\}$ (*e.g.,* using the output of the VDF as the random seed for a pre-agreed pseudorandom number generator). VDF is necessary to prevent the DP that lastly publishes its nonce from introducing bias by strategically selecting its nonce.

After random sampling, the public input and private witness for the proving scheme should be also adjusted accordingly as $\mathbb{W}^{t,c}_g = \{\mathbb{W}^t_{g,r^t_1}, \mathbb{W}^t_{g,r^t_2}, \ldots, \mathbb{W}^t_{g,r^t_c}\}, \mathbb{W}^{t-1,c}_g = \{\mathbb{W}^{t-1}_{g,r^t_1}, \mathbb{W}^{t-1}_{g,r^t_2}, \ldots, \mathbb{W}^{t-1}_{g,r^t_c}\}, \mathbb{U}^{t,c} = \{u^t_{r^t_1}, u^t_{r^t_2}, \ldots, u^t_{r^t_c}\}$, where $\mathbb{W}^{t,c}_g \in \mathbb{R}^{1\times c}$, $\mathbb{W}^{t-1,c}_g \in \mathbb{R}^{1\times c}$, $\mathbb{U}^{t,c} \in \mathbb{R}^{n\times c}$. As a prerequisite for using the sample-based verification, the DA shall publish the model $\mathbb{W}^t_g$ before sampling (since only part of the $\mathbb{W}^t_g$ is used as the public input).

5.1.4 *Integrated Verification Protocol*

Taken all parts together, our verifiable aggregation protocol proceeds as follows. The DA first quantizes W^{t-1}_g, U^t and K^t to the quantized format, and performs the aggregation calculation as Equation (4) and Equation (6). In addition, the DA applies the dequantization equation to calculate the floating-point global model W^t_g, and commits K^t, W^t_g and $\mathbb{W}^t_g$ to DPs. Afterwards, the DA obtains the randomly selected parameters from the VDF, and generates a zero-knowledge proof π^t with public input as $\mathbb{X}^{t,c} = \{\mathbb{W}^{t,c}_g, \mathbb{W}^{t-1,c}_g, \mathbb{K}^t\}$ and private witness as $\mathbb{W}^{t,c} = \{\mathbb{U}^{t,c}\}$. The proof π^t is then submitted to a smart contract so that the DPs can verify its correctness and claim corresponding rewards (see § 5.2).

5.2 The Trading Smart Contract

martFL designs a trading smart contract to enable the DA and DPs to exchange plaintext local models and rewards. Due to space constraint, we provide the high-level description of our smart contract in Algorithm 2. The more detailed realization close to the real-world implementation is deferred to the technical report [49]. The trading smart contract is divided into two high-level phases.

Prepare Phase. The PreparePhase performs necessary setup for reward distribution. First, the DA commits the aggregation weights K^t and the corresponding DPs (identified by their public keys or addresses on blockchain) in the smart contract. Meanwhile, the DA deposits the reward v_{DPs} for the DPs proportional to their weights in K^t. Additionally, the DA also deposits v_{DA} as the penalty if it cannot later provide a correct proof. Afterwards, the DPs can safely submit their plaintext local models off-chain to the DA, based on which the DA generates the verifiable proof as described in § 5.1. After proof generation, the DA commits the public inputs.

Algorithm 2: The Trading Smart Contract

```
PreparePhase() :
    commit K^t, addrs # addrs identify selected DPs in current epoch
    v_DPs, v_DA = Deposit(msg.value) # DA deposits (reward, penalty)
    R_DPs := Allocate(v_DPs, K^t) # allocate DPs reward based on K^t
    U^t := Submission(addrs) # DPs submit local models off-chain
    W_g^t := Aggregate(W_g^{t-1}, K^t, U^t) # DA generates proof off-chain
    commit W_g^t, W_g^{t-1}, K^t # DA commits public inputs
VerifyPhase() :
    DA performs verifiable sampling offline and publishes s^t_vdf, π^t_vdf
    DA adjusts W_g^{t,c}, W_g^{t-1,c} and U^{t,c} based on R^t_vdf
    DA generates π^t_agg :=Prove(X^{t,c}, W^{t,c}, pk, C) offline
    DA publishes the proof π^t_agg on-chain
    DPs invokes verification v^t := Verify(vk, X^{t,c}, π^t_agg)
    if v^t = false :
        distribute both the security deposit v_DA the award v_DPs to DPs
    else : distribute v_DPs to DPs and return v_DA to DA
Function Verify(vk, X^{t,c}, π^t_agg) :
    s := Σ_{i=0}^{Len(X^{t,c})-1} ScalarMul(vk.γ_abc[i + 1], X^{t,c}[i])
    s :=Addition(s, vk.γ_abc[0])
    p1 := π^t_agg.a, Negate(s), Negate(π^t_agg.c), Negate(vk.α)
    p2 := π^t_agg.b, vk.γ, vk.δ, vk.β
    return PairingCheck(p1,p2)
```

Verify Phase. The second phase focuses on verifying the integrity of model aggregation. The DA performs verifiable random sampling and provides the proper proof (*i.e.,* π^t_{vdf}) for randomness. Afterwards, the DA adjusts the public and private inputs according to the random seed s^t_{vdf}, based on which it generates the final proof for model aggregation π^t_{agg}. The proof is uploaded to the trading smart contract such that any DP can verify its correctness by invoking the on-chain Verify function. The DA will lose its security deposit if the verification fails.

On-Chain Verification Procedure. The Verify function is responsible for checking the correctness of π^t_{agg}. It takes input as the committed verification key vk and the quantized public input $\mathbb{X}^{t,c}$, and the proof π^t_{agg}. The underlying verification is based on the Groth16 protocol [37] which checks four pairings. The cryptography-related computations (such as Addition and ScalarMul) are implemented via the precompiled smart contracts to reduce gas cost.

6 EVALUATION

6.1 Experimental Setup

Our experiments are conducted on two Linux servers with Intel(R) Xeon(R) Gold 6348 CPU and NVIDIA RTX A100 GPU. We use Pytorch [61] to implement FL, apply SEAL [1] for CKKS-based Homomorphic operations, and Ethereum [68] testnet for deploying our trading smart contract. The source code is available at Github[2]. All results are obtained based on five repetitions of experiments.

[2]https://github.com/liqi16/martFL

Datasets, Models, and Baselines. We use multiple datasets from different domains in our evaluations, including two image classification datasets, FMNIST [69] and CIFAR [46], and two text classification datasets, TREC [50] and AGNEWS [75]. We train LeNet [47] as global model for FMNIST [69], TextCNN [73] for TREC [50] and AGNEWS [75]. We train a convolutional neural network (CNN) with three CNN layers and four linear layers as the global model for the CIFAR [46] dataset. We compare martFL with two server-driven methods (FLTrust [18] and CFFL [56]) and four client-driven methods (FedAvg [58], RFFL [71], Krum [12], and Median [72]).

Training. We set the same number of participants for both client-driven and server-driven approaches. This ensures that all participants use the same number of samples in the training process. For client-driven methods, we set n DPs. For server-driven methods, we set one DA and $n-1$ DPs. For image classification tasks, we set 30 participants, the optimizer is SGD, and the learning rate is 1.0×10^{-2}. For text classification tasks, we set 20 participants, the optimizer is Adam, and the learning rate is 5.0×10^{-5}. The number of samples in the DA's root dataset is 200 for FMNIST, CIFAR, and AGNEWS, and 120 for TREC, counting for roughly 0.3%, 0.4%, 2%, and 1.6% of the total data held by the DPs, respectively. Unless otherwise specified, we train a model until its peak accuracy on our validation dataset does not increase for 100 training epochs.

Data Splits. We apply two sampling methods to divide the amount of data held by each DP: UNI and POW. In UNI, each DP has the same amount of samples; in POW method, the numbers of samples owned by different DPs follow a power-law distribution. In addition, we divide the local data distribution of the DPs according to two methods, IID and NonIID. IID means that each DP has all classes of samples and the samples in each class are uniformly distributed; NonIID means that the DP has a subset of classes, and the data distributions vary for different DPs.

The Adversary. We consider two untargeted attacks, two targeted attacks, and Sybil attack [31]. The untargeted attacks include sign-randomizing attack and free-rider attack [52]. The sign-randomizing attack is an attack on the direction of the gradients where the adversary randomly sets the sign as +1 or −1. In the free-rider attack, we implement the delta weight attack [52], which generates gradient updates by subtracting the two global models received in the previous two epochs. The targeted attacks include label-flipping attacks and backdoor attacks [9]. In a label-flipping attack, the adversary swaps the labels of the two classes of data in the training process to train poisoned local models. In the Sybil attack, the adversary conjures up a number of clients and submit the same compromised model. In the Sybil attack, we use the label-flipping attack to train the malicious local models.

Evaluation Metrics. We use Main Task Accuracy (**MTA**) and Attack Success Rate (**ASR**) as the evaluation metrics. MTA measures the classification accuracies of the trained models, while ASR measures the fraction of poisoned samples that are predicted as the target class in targeted attacks. Thus, higher MTAs indicate more effective models, and lower ASRs indicate more robust models against targeted attacks. We further define Data Acquisition Cost (**DAC**) as the average percentage of local models that the DA must procure in each training epoch in order to train the global model. In general, the DA seeks to obtain high-performing models (*i.e.,* with high MTAs and low ASRs) at a reasonable DAC (lower the better).

Default Hyper-Parameters. For martFL, we set the threshold T used in hierarchical clustering as 0.05 and the ratio of randomly selected baseline candidates β as 0.1. For Krum [12], we set the proportion of possibly Byzantine as 20%. For CFFL [56], we set the coefficient of reputation threshold as 1.0 and α as 5. For RFFL [71], we set the hyper-parameter α as 0.95 and threshold as 1.0. For the backdoor attack, we implement the attack proposed in [9] where the hyper-parameter α is 0.95.

6.2 Evaluation Results

Our evaluations are centered around the following questions:

- **Accuracy**. In § 6.2.1 and § 6.2.2, we quantitatively show that martFL achieves the best MTAs compared to other server-driven methods regardless of when the DA's root dataset is biased or not. Meanwhile, martFL reduces up to 69% DAC when achieving comparable (if not better) MTAs with prior arts.
- **Robustness.** In § 6.2.3, we show that when facing with various targeted attacks, untargeted attacks, and Sybil attack, martFL can accurately identify malicious DPs and achieve the highest MTA and lowest ASR in most cases, compared with prior arts.
- **Accuracy Loss by Quantization.** In § 6.2.4, we show that quantization has little to no impact on the MTA of the global model.
- **System Overhead**. In § 6.2.5, we study the system overhead of martFL, including the cryptography overhead during local model evaluations, and the gas cost incurred for executing the trading smart contract.

6.2.1 *Biased Root Dataset*

First, we evaluate the MTA of prior art when the DA possesses an unevenly distributed root dataset. Specifically, we consider that (i) the root dataset of DA is dominated by half of the class labels; (ii) the DPs follow the form of UNI in the number of samples; (iii) a certain percentage of DPs have biased local dataset and the remaining DPs have evenly distributed dataset (*i.e.,* high-quality dataset with IID distributions across all class labels).

We evaluate three different percentages of DPs (20%, 30%, and 40%) possessing biased local datasets. The results are reported in Table 1. In terms of main task accuracy (MTA), martFL consistently outperforms existing server-driven approaches, with particularly significant advantages over FLTrust. We observed that all three methods have very close MTAs on the FMNIST task. This may be because the FMNIST task is relatively simple and we use a fairly small model with approximately 44,000 parameters. The advantages of martFL become more pronounced on larger models (for instance, the models for both text classification tasks have ∼3 million parameters, and the model for the CIFAR task has ∼1 million parameters). The underlying reason for the MTA improvements in martFL is because existing server-driven approaches has poor inclusiveness when the root dataset is biased. To quantify this, we plot the Cumulative Distribution Function (CDF) of inclusiveness in Figure 5 for the TREC task with 20% biased DPs. We consider both the inclusiveness of all the DPs and inclusiveness of only the high-quality DPs. Because existing server-driven methods tend to select local models with data distributions similar to the DA's root

Dataset	Biased Ratio	20%		30%		40%	
	Metric	MTA	DAC	MTA	DAC	MTA	DAC
TREC	CFFL	76.87 ± 6.87	100.00	81.53 ± 0.90	100.00	79.07 ± 3.24	100.00
	FLTrust	67.40 ± 4.76	36.52	72.60 ± 1.07	46.47	71.73 ± 1.09	40.15
	Ours	**88.80 ± 1.72**	53.63	**87.53 ± 1.15**	53.88	**87.20 ± 0.49**	51.79
AGNEWS	CFFL	44.09 ± 1.95	100.00	43.39 ± 1.57	100.00	45.58 ± 1.46	100.00
	FLTrust	44.09 ± 1.43	11.52	45.19 ± 0.39	10.35	43.65 ± 0.90	11.89
	Ours	**79.71 ± 2.15**	36.30	**75.89 ± 1.30**	38.99	**78.04 ± 1.08**	41.15
FMNIST	CFFL	**88.37 ± 0.55**	100.00	88.48 ± 0.25	100.00	**88.02 ± 0.32**	100.00
	FLTrust	87.33 ± 0.48	32.64	87.26 ± 0.38	33.57	87.28± 0.39	46.04
	Ours	88.22 ± 0.26	35.14	**88.88 ± 0.27**	30.06	87.71 ± 0.43	39.60
CIFAR	CFFL	63.34 ± 0.22	100.00	62.38 ± 0.33	100.00	60.85 ± 0.80	100.00
	FLTrust	10.00 ± 0.00	7.59	11.42 ± 1.00	10.79	14.25 ± 1.99	1.30
	Ours	**64.24 ± 0.06**	53.63	**63.79 ± 0.28**	53.88	**62.60 ± 0.37**	51.79

Table 1: MTA (%) and DAC (%) when the DA possesses a biased root dataset.

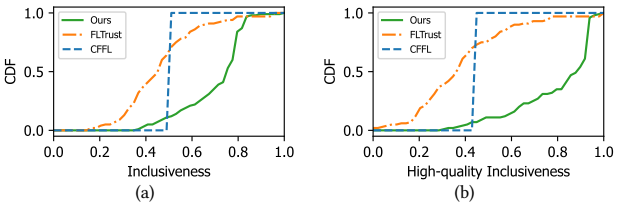


Figure 5: The inclusiveness analysis when the DA posseses a biased root dataset.

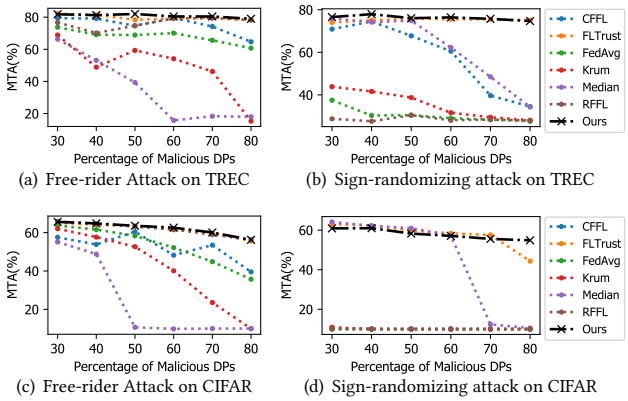


Figure 6: The MTA of the global model obtained by different aggregation protocols under untargeted attacks.

dataset, their selection of DPs is highly biased towards its root dataset. On the contrary, benefited from the dynamic baseline adjustment design, martFL can include more high-quality DPs, even if the root dataset is biased.

We further report DACs for all three methods, which represents the average percentage of local models that the DA purchases in each training epoch. The DAC in CFFL is always 100% because CFFL must obtain all local models and evaluate their accuracies before deciding whether or not to aggregate them. Therefore, the model aggregation design in CFFL is undesirable in data marketplace, where the DA prefers to only pay for high-quality local models from the DPs. On the contrary, FLTrust has low DACs in this setting because its local model selections are highly biased. As a result, FLTrust has the lowest MTAs in nearly all tasks. martFL instead strikes a good balance between MTA and DAC, allowing the DA to obtain high-performing global models with low cost.

6.2.2 *Unbiased Root Dataset*

In this segment, we evaluate the scenario where the DA's root dataset is unbiased. The total number of data samples owned by each DP follows the POW distribution. However, each DP has evenly distributed class labels. The results are shown in Table 2. In general, when the root dataset is reliable, all three methods have better MTAs than the case where the root dataset is biased. martFL achieves slightly better or comparable MTAs compared with other methods with the lowest DACs in all four tasks.

With the results in Table 1 and Table 2, we demonstrate that (i) CFFL is slightly more resilient against a biased root dataset than FLTrust. Yet, CFFL introduces consistently high DACs, which is less desirable in data marketplace. (ii) FLTrust, on the other hand, heavily depends on the root dataset, and can only achieve comparable MTAs with CFFL when the root dataset is unbiased. In contrast, martFL produces the best MTAs in nearly all cases regardless of whether the root dataset is biased or not. Crucially, martFL maintains the lowest DACs when achieving comparable MTAs with the other two methods.

6.2.3 *Robustness Against Various Attacks*

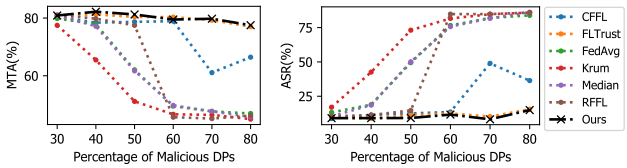
In this case, we consider the robustness of martFL when facing malicious DPs. We compare martFL with both client-driven and server-driven approaches. Since we investigate over nearly 600 different combinations of approaches, attacks, and tasks, we train each combination for a fixed number of 100 epochs in this segment.

First, Figure 6 presents the MTA of each scheme under free-rider attack and sign-randomizing attack on the TREC and CIFAR dataset. The result shows that martFL can defend against the attacks even 80% of the DPs are malicious. For the free-rider attack, the MTA of martFL slightly decreases by 2.80% when the number of faulty DPs increases from 30% to 80%. For the sign-randomizing attack, the MTA of martFL remains consistent given different numbers of faulty DPs.

Second, we plot the robustness of different aggregation schemes against targeted attacks on the TREC and CIFAR dataset in Figure

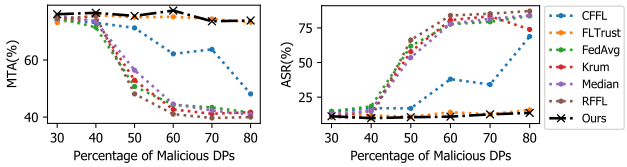
Dataset	TREC		AGNEWS		FMNIST		CIFAR	
Metric	MTA	DAC	MTA	DAC	MTA	DAC	MTA	DAC
CFFL	85.47 ± 0.68	100.00	78.79 ± 1.03	100.00	89.22 ± 0.15	100.00	65.38 ± 0.50	100.00
FLTrust	87.40 ± 0.71	46.65	80.94 ± 1.26	66.11	89.40 ± 0.20	51.62	**70.66 ± 0.45**	39.44
Ours	**87.67 ± 0.57**	44.38	**83.35 ± 1.54**	65.61	**89.88 ± 0.15**	45.27	70.28 ± 0.27	34.89

Table 2: MTA (%) and DAC (%) when the DA has an unbiased root dataset.



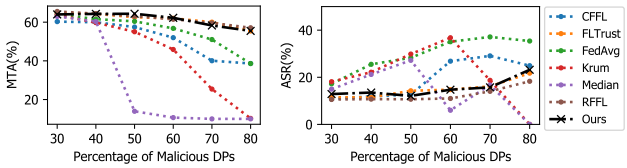
(a) MTA(%) against Backdoor Attack on TREC

(b) ASR(%) against Backdoor Attack on TREC



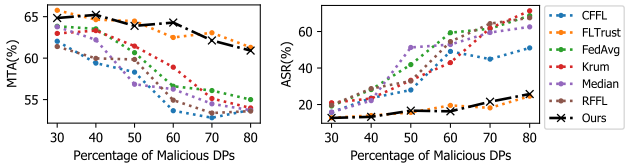
(c) MTA(%) against Label-Flipping Attack on TREC

(d) ASR(%) against Label-Flipping Attack on TREC



(e) MTA(%) against Backdoor Attack on CIFAR

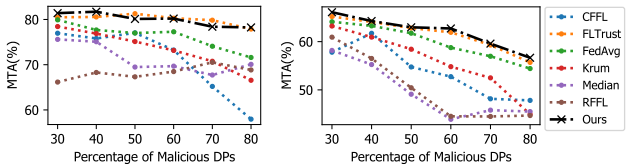
(f) ASR(%) against Backdoor Attack on CIFAR



(g) MTA(%) against Label-Flipping Attack on CIFAR

(h) ASR(%) against Label-Flipping Attack on CIFAR

Figure 7: The MTA and ASR of the global model obtained by different aggregation protocols under targeted attacks.



(a) TREC (b) CIFAR

Figure 8: The MTA of the global model obtained by different aggregation protocols against Sybil attacks.

Dataset	ATK	Ratio	30%	40%	50%	60%	70%	80%
TREC	FR	FLTrust	99.22	84.38	**36.18**	**31.4**	24.22	13.68
		Ours	**87.32**	**68.32**	46.58	31.54	**20.26**	**12.02**
	SR	FLTrust	**61.64**	**53.32**	42.82	**33.50**	26.18	**16.06**
		Ours	67.44	57.28	**40.3**	37.98	**22.82**	17.4
	BD	FLTrust	80.54	68.60	72.24	67.38	53.72	45.62
		Ours	**53.76**	**51.50**	**46.60**	**39.60**	**29.64**	**41.86**
	LF	FLTrust	83.06	81.24	65.28	70.66	58.04	53.46
		Ours	**61.12**	**52.40**	**47.92**	**56.44**	**30.02**	**24.96**
	SY	FLTrust	67.20	62.64	66.72	66.98	70.40	77.92
		Ours	**53.76**	**48.10**	**48.68**	**44.9**	**50.62**	**60.66**
CIFAR	FR	FLTrust	72.22	63.24	51.09	48.56	35.82	**20.71**
		Ours	**68.02**	**60.36**	**45.91**	**46.49**	**33.02**	22.13
	SR	FLTrust	**59.76**	**53.82**	**47.24**	**40.29**	**33.33**	**25.93**
		Ours	76.73	63.38	57.6	52.18	46.16	30.73
	BD	FLTrust	72.29	72.20	68.27	73.09	69.67	73.64
		Ours	**60.93**	**60.64**	**54.44**	**54.56**	**70.62**	**49.27**
	LF	FLTrust	74.62	74.38	72.69	72.76	70.42	66.42
		Ours	**65.98**	**60.73**	**63.07**	**56.27**	**54.42**	**44.58**
	SY	FLTrust	61.29	65.38	68.42	70.2	64.38	61.33
		Ours	**56.00**	**50.53**	**48.27**	**44.2**	**40.36**	**39.33**

* In this table, "ATK" represents the type of attacks, "FR" represents the free-rider attack, "SR" represents the sign-randomizing attack, "BD" represents the backdoor attack, "LF" represents the label-flipping attack, and "SY" represents the Sybil attack.

Table 3: The DAC (%) comparison between FLTrust and martFL in the robustness experiments. Results for CFFL are omitted since they are always 100.

7. The results show that martFL can achieve the highest MTA and lowest ASR in most cases. Note that in Figure 7(f), the ASR of Krum and Median initially increase, but then decrease to 0. However, the MTAs of both methods also decrease to zero, as shown in Figure 7(e). This indicates that the global model is not converged for both methods when the percentage of malicious DPs is over 60%.

Finally, we evaluate the robustness of different schemes against the Sybil attack in Figure 8. The experimental results show martFL and FLTrust have comparable MTAs in most cases. In contrast, the MTA of other methods decrease significantly as the number of Sybil nodes increases. This is because the models submitted by Sybil nodes are similar to each other, so that these schemes cannot accurately eliminate the poisoned local model updates.

To sum up, CFFL and all the client-driven methods are more vulnerable to faulty DPs. For instance, some client-driven aggregation protocols cannot produce a meaningful global model when the number of malicious DPs is sufficiently large. martFL achieves slightly better robustness than FLTrust, yet reducing roughly 13.91% DAC on average compared with FLTrust, as shown in Table 3.

6.2.4 *Accuracy Loss by Quantization*

To enable verifiable data transaction in martFL, the DA needs to first quantize the prior model W_g^{t-1} and local model updates U^t, perform the aggregation to obtain quantized global model $\mathbb{W}_g^t$, and then de-quantize the model to obtain a floating-point model W_g^t. In this segment, we study the impact of quantization on model accuracy. We evaluate the scenario where the DA's root dataset is unbiased.

Dataset	MTA		F1	
	QT	ΔQT	QT	ΔQT
TREC	88.60 ± 0.28	-0.93 ↑	87.58 ± 0.37	-0.65 ↑
AGNEWS	82.61 ± 0.65	0.74 ↓	82.54 ± 0.65	0.64 ↓
FMNIST	90.07 ± 0.07	-0.20 ↑	90.03 ± 0.07	-0.20 ↑
CIFAR	69.74 ± 0.62	0.54 ↓	69.74 ± 0.69	0.53 ↓

Table 4: The MTA(%) and F1(%) loss in quantization.

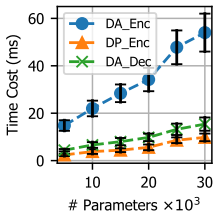


Figure 9: The homomorphic ENC/DEC times.

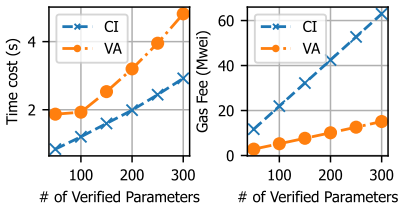


Figure 10: The gas cost and execution times for verifying different numbers of parameters.

Phrase	Prepare				Verify		Any
Function	NE	Deposit	CM	Prepare	CR(DA)	CR(DP)	RE
Gas (wei)	163076	46643	127731	222018	38036	42632	-
Time (ms)	92.0	73.6	80.6	83.4	64.2	70.6	74.8

* In this table, "NE" represents NewEpoch, "CM" represents CommitModel,"CR" represents ClaimReward, and "RE" represents ReadEpoch.

Table 5: The gas costs and execution times of the functions in our trading smart contract.

The total number of data samples and the type of labels owned by each DP follows the POW and IID distribution, respectively. The results are summarized in Table 4. The ΔQT represents the MTA and F1 loss due to quantization (*i.e.,* the accuracy difference between W_g^t and $\mathbb{W}_g^t$). The results indicate that the quantization operations introduce negligible accuracy losses.

6.2.5 *System-level Overhead*

In this segment, we study the system-level overhead of (i), including the cryptography overhead in our quality-aware model evaluation protocol; (ii) the time and gas cost of the functions in the trading smart contract.

In Figure 9, we plot the overhead of homomorphic encryption and decryption operations. In the experiment, we set 4096 slots per batch for the CKKS algorithm. The encryption time of the DA is longer because the DA needs to perform homomorphic encryption, while DPs only needs to complete homomorphic additions. Overall, the extra latency introduced by homomorphic operations is small.

Further, we report the gas cost and latency for executing different functions in our trading smart contract in § 5.2. We developed a set of key functions in the smart contract (see detailed implementations in [49]). In the PreparePhase, NewEpoch initializes training epochs, Deposit enables the DA to deposit rewards and penalties. CommitModel allows the DPs to commit local models, and Prepare records rewards and selects verification parameters. In the VerifyPhase, CommitInputs (abbreviated as "CI") allows the DA to commit the public inputs for aggregation verification, and VerifyAggregation (abbreviated as "VA") verifies the integrity of aggregation. Both the DA and DPs can use ClaimReward to claim rewards. Finally, ReadEpoch is a convenience handle to return detailed epoch information. Overall, none of these functions consumes more than 0.25×10^6 wei, which costs less than 0.0025 US dollars at the time of this writing.

In comparison to Omnilytics [51], which directly aggregates local models via a smart contract, the gas cost in martFL is at least 1000 times smaller, when enabling approximately 8 times more participants to collectively train models with ~100 times more parameters than the model trained in Omnilytics. In fact, the gas cost in martFL is independent of the model size and the model evaluation method, since martFL only verifies a fixed number (a system setting) of randomly sampled model parameters. In addition, the execution time of each function is less than 0.1 seconds. In Figure 10, we also report the gas cost and execution time when martFL has different system settings that verify different numbers of model parameters.

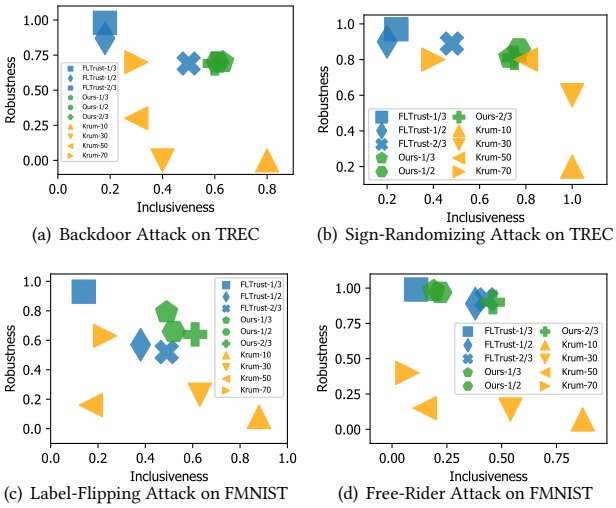
6.3 Deep Dive

In this section, we further investigate several key design choices of martFL and outline some future work.

6.3.1 *Tradeoff Between Inclusiveness and Robustness*

In § 4.1, we presented the key observation that existing approaches exhibit a fundamental tradeoff between inclusiveness and robustness when aggregating local models submitted by the DPs. In this segment, we further analyze this tradeoff by tuning the key parameters in the aggregation algorithms of both FLTrust [18] and Krum [12]. Specifically, since FLTrust selects local models based on the cosine similarities between them and the self-computed model update trained on the DA's root dataset, the key system parameter that dictates the aggregation in FLTrust is the quality of the root dataset. We quantify the quality as *unbiasness ratio*, which represents the percentage of class labels that predominates the DA's root dataset. For instance, FLTrust-1/3 represents the case where the root dataset contains 1/3 of the class labels. In Krum, the key tunable parameter f is the proportion of Byzantine DPs defined in Krum's problem formulation, which directly determines the number of local models selected for aggregation in each epoch. We evaluate four Krum settings in this part (for instance, Krum-50 represents the setting where the Krum algorithm is supposed to tolerate 50% Byzantine DPs). We consider a group of heterogeneous DPs, where 30% of them hold high-quality data (evenly distributed across all types of labels), 30% of them hold biased datasets in which the class labels are dominated by half of the randomly selected labels, and 40% of them are malicious. We experiment both the targeted and untargeted attacks on the TREC and FMNIST tasks.

The results are plotted in Figure 11, where inclusiveness is the percentage of benign DPs whose local models are selected for aggregation, and robustness is quantified as the percentage of malicious DPs that are excluded for aggregation. We collect these two metrics in each training epoch and report the average values. In general, FLTrust can achieve reasonably good inclusiveness only if the unbiasness ratio of the DA's root dataset is sufficiently large (*e.g.,* reaching 2/3). Krum, which is significantly impacted by its Byzantine tolerance threshold f, tends to have high robustness at the expense



(a) Backdoor Attack on TREC (b) Sign-Randomizing Attack on TREC

(c) Label-Flipping Attack on FMNIST (d) Free-Rider Attack on FMNIST

Figure 11: Deep dive into the tradeoff between inclusiveness and robustness in various settings.

Dataset	TREC			FMNIST		
Attack	BD		SR	LF		FR
Metric	MTA	ASR	MTA	MTA	ASR	MTA
FLTrust-1/3	48.60	12.02	57.08	74.71	18.96	70.71
FLTrust-1/2	58.84	9.01	57.48	87.11	4.69	86.52
FLTrust-2/3	81.53	9.43	83.16	87.97	5.37	86.38
Krum-10	70.11	41.58	32.08	88.43	5.12	87.69
Krum-30	61.64	68.27	38.24	86.44	13.43	86.60
Krum-50	64.16	67.04	**85.08**	80.64	32.20	79.16
Krum-70	72.64	36.23	78.64	85.97	4.95	8.19
Ours-1/3	83.71	7.75	76.36	87.94	**3.41**	81.19
Ours-1/2	83.38	8.43	75.36	88.23	4.48	84.34
Ours-2/3	**84.08**	**7.55**	79.96	**88.65**	3.94	**88.18**

* In this table, "BD" represents the backdoor attack, "SR" represents the sign-randomizing attack, "LF" represents the label-flipping attack, and "FR" represents the free-rider attack.

Table 6: The MTA (%) and ASR (%) in the inclusiveness-robustness tradeoff experiments.

of inclusiveness. In contrast, martFL achieves the best tradeoff between inclusiveness and robustness in all cases. As a result, martFL consistently achieves high model performance regardless of the parameter settings, as summarized in Table 6. However, FLTrust begins to match martFL only when the root dataset is sufficiently good. Krum fails to perform consistently across all tasks, regardless of the parameter choices. For instance, although Krum-50 achieves a good MTA on TREC against the sign-randomizing attack, it has a significantly high ASR against the backdoor attack.

6.3.2 *Analysis of Dynamic Baseline Adjustment*

Quantitative Results. We first quantitatively analyze the accuracy of our Dynamic Baseline Adjustment algorithm (detailed in § 4.2.2) to demonstrate its robustness in various data distributions. We consider a challenging setting for the DPs, where 30% of them hold high-quality data and 30% of them hold biased datasets. The remaining 40% of DPs have three different settings: (i) holding evenly-distributed data, (ii) maliciously engaging the backdoor attack (one type of targeted attack), and (iii) maliciously engaging the sign-randomizing attack (one type of untargeted attack). In terms of the root dataset held by the DA, we consider the following three scenarios:

- **Type-I Bias.** The root dataset of DA is dominated by half of the class labels. The data distributions of the biased DPs are similar to the distributions of the DA's root dataset.
- **Type-II Bias.** The root dataset of DA is dominated by half of the class labels. However, the data distributions of the biased DPs are random and independent of the DA's data distributions.
- **Unbiased.** The root dataset of DA is evenly distributed.

We analyze the probability of selecting correct baselines during the training process. A baseline is correct if the cosine similarity between the baseline and the *ground truth model update* is strictly positive. The ground truth update is an ideal update obtained directly on high-quality data, which can be considered as *a hypothetical scenario* where the DA possesses sufficient high-quality data and can train the model all by itself.

The results are reported in Table 7. The Type-I Bias is arguably the most challenging scenario because the DA and the biased DPs are *similarly biased*. Therefore, the DA is prone to be misled by these biased DPs, resulting in possible incorrect baseline selections (note that all our experiments in § 6.2 considered the Type-I Bias). Nonetheless, our method still achieves reasonably accurate baseline selections, up to 99% accuracy in the training epochs of the CIFRA task. We also observed that the selection accuracies are affected by the total number of class labels. Specifically, TREC and AGNEWS have 6 and 4 class labels, respectively, while FMNIST and CIFAR both have 10 class labels. When the total number of class labels is smaller, the data diversity experienced by the biased DA and DPs is even lower (for instance, they only see 2 labels in the AGNEWS task). This results in a relatively higher probability of selecting incorrect baselines. In the Type-II Bias scenario, where the data distributions of the biased DPs and the DA are not correlated, the probabilities of selecting correct baselines are higher than the Type-I Bias scenario, even for the TREC and AGNEWS task. In fact, the selection accuracies in the Type-II Bias scenario are already comparable to the case where the DA's root dataset is unbiased, achieving over 90.0% baseline selection accuracy in most settings.

Attack	Scenarios	TREC	AGNEWS	FMNIST	CIFAR
None	Type-I Biased	80.00	70.00	98.80	95.67
	Type-II Biased	88.33	98.67	100.00	100.00
	Unbiased	95.67	100.00	99.67	99.67
BD	Type-I Biased	65.33	58.00	96.67	95.00
	Type-II Biased	80.33	93.00	95.33	100.00
	Unbiased	79.00	84.00	100.00	100.00
SR	Type-I Biased	64.67	62.67	98.67	99.00
	Type-II Biased	96.00	88.67	99.00	89.50
	Unbiased	95.67	99.67	99.33	91.50

"BD" and "SR" represent the backdoor and sign-randomizing attack, respectively.

Table 7: The probability of selecting correct baselines by our dynamic baseline adjustment algorithm.

Attack Discussion. To attack our model aggregation protocol, an adversary must carefully design local models that can be selected as the baseline. However, there are several challenges to crafting such models. First, the adversary has no access to the plaintext models submitted by other DPs throughout the model aggregation and transaction. Instead, the adversary only observes the commitments of these local models, which do not reveal the actual local models. Additionally, because the DA evaluates all local models offline using the private model evaluation algorithm with its private root dataset, it is difficult for the adversary to predict what types of local models will receive higher Kappa scores in the root dataset. Finally, the DA discloses the DP that is selected as the baseline for the current epoch only after all DPs have committed their models. At this point, the adversary cannot modify its committed local model, even if it colludes with the selected DP. In summary, because martFL enables complete offline and private local model evaluation and aggregation by the DA, attacking our aggregation protocol is significantly more difficult than existing blockchain-based approaches (*e.g.,* Omnilytics [51], FPPDL [57]) that require the aggregation protocol to be publicly observable on the blockchain.

Future Work on Model Evaluation and Aggregation Protocols. Several studies [8, 40] have shown that historical information can be used to identify Byzantine DPs. In our future work, we intend to leverage this insight by taking into account DPs' historical reputations and incorporate momentum into our local model evaluation algorithm. This approach may further reduce the reliance on the DA's root dataset and compensate for errors in baseline selections, particularly in the Type-I Bias scenario.

7 RELATED WORK

Verifiable Protocols for Real-world Systems. The enthusiasm for Web 3.0 [54] drives a growing number of literature on empowering or even transforming real-world systems via verifiable (or trust-free) protocols, such as verifiable cloud computing (*e.g.,* [24, 48]), decentralized digital good exchanges (*e.g.,* [25]), smart contract based legal sector transformation [28], and various proposals to improve the Blockchain systems themselves (such as interoperability (*e.g.,* [53, 70, 74]) and private smart contracting [20, 41]). Overall, practicability and deployability are two the primary challenges for designing verifiable protocols to power real-world systems. Thus, the verifiable transaction protocol in martFL only proves the critical computations that are necessary and sufficient to ensure fair billing. It does not verify the entire FL training process, which would otherwise impose unacceptably high overhead.

Data Pricing. Prior works have studied data pricing. For instance, proposals [33, 39] evaluate data value based on Shapley Value using a game theoretic approach, which typically requires access to the full datasets. Some other literatures (*e.g.,* [10, 43]) propose a pricing framework for relational queries. Data pricing for FL-based marketplace is part of our future work.

8 CONCLUSION

In this paper, we propose martFL, a novel FL architecture that is specifically designed to enable a utility-driven data marketplace. Benefiting from the quality-aware model evaluation protocol, martFL can eliminate the tradeoff between inclusiveness and robustness when selecting desired DPs. Further, martFL designs a verifiable transaction protocol that enables the DA to prove that it faithfully aggregates model using the committed aggregations weights, enabling fair trading between the DA and DPs. We implemented a prototype of martFL and extensively evaluated it on four datasets. The experimental results demonstrate the accuracy, robustness and efficiency of martFL.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. The research is supported in part by the National Key R&D Program of China under Grant 2022YFB2403900, NSFC under Grant 62132011, and China National Funds for Distinguished Young Scientists under Grant 61825204.

REFERENCES

[1] Microsoft SEAL (release 3.7.2). https://github.com/Microsoft/SEAL, 2019.
[2] Libsnark: C++ Library for zkSNARKs. https://github.com/scipr-lab/libsnark, 2020.
[3] Bdex. https://www.bdex.com/, 2022.
[4] Arkworks-rs/Groth16: A Rust Implementation of the Groth16 zkSNARK. https://github.com/arkworks-rs/groth16, 2023.
[5] GE Predix Platform: Industrial IoT Platform: GE Digital. https://www.ge.com/digital/iiot-platform, 2023.
[6] International Data Spaces Association. https://internationaldataspaces.org/, 2023.
[7] Quandl. https://demo.quandl.com/, 2023.
[8] Allen-Zhu, Z., Ebrahimian, F., Li, J., and Alistarh, D. Byzantine-resilient Non-convex Stochastic Gradient Descent. In *International Conference on Learning Representations (ICLR)* (2021).
[9] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to Backdoor Federated Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2020).
[10] Balazinska, M., Howe, B., and Suciu, D. Data Markets in the Cloud: An Opportunity for the Database Community. *VLDB Endowment* (2011).
[11] Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. signSGD with Majority Vote is Communication Efficient and Fault Tolerant. In *International Conference on Learning Representations (ICLR)* (2019).
[12] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems (NeurIPS)* (2017).
[13] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical Secure Aggregation for Privacy-preserving Machine Learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2017).
[14] Boneh, D., Bonneau, J., Bünz, B., and Fisch, B. Verifiable Delay Functions. In *Annual International Cryptology Conference (CRYPTO)* (2018).
[15] Bowe, S., Grigg, J., and Hopwood, D. Recursive Proof Composition without a Trusted Setup. In *Cryptology ePrint Archive* (2019).
[16] Bünz, B., Fisch, B., and Szepieniec, A. Transparent SNARKs from DARK Compilers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)* (2020).
[17] Calzada, I. Citizens's Data Privacy in China: The State of the Art of the Personal Information Protection Law (PIPL). In *Smart Cities* (2022).
[18] Cao, X., Fang, M., Liu, J., and Gong, N. Z. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *ISOC Network and Distributed System Security Symposium (NDSS)* (2021).
[19] Chen, H., Asif, S. A., Park, J., Shen, C.-C., and Bennis, M. Robust Blockchained Federated Learning with Model Validation and Proof-of-Stake Inspired Consensus. In *AAAI Conference on Artificial Intelligence (AAAI) Workshop on Towards Robust, Secure and Efficient Machine Learning* (2021).
[20] Cheng, R., Zhang, F., Kos, J., He, W., Hynes, N., Johnson, N., Juels, A., Miller, A., and Song, D. Ekiden: A Platform for Confidentiality-preserving, Trustworthy, and Performant Smart Contracts. In *IEEE European Symposium on Security and Privacy* (2019).
[21] Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)* (2017).
[22] Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., and Ward, N. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory*

and Applications of Cryptographic Techniques (EUROCRYPT) (2020).

[23] Cohen, J. A coefficient of agreement for nominal scales. In *Educational and psychological measurement* (1960).

[24] Dong, C., Wang, Y., Aldweesh, A., McCorry, P., and van Moorsel, A. Betrayal, Distrust, and Rationality: Smart Counter-Collusion Contracts for Verifiable Cloud Computing. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security (CCS)* (2017).

[25] Dziembowski, S., Eckey, L., and Faust, S. Fairswap: How to Fairly Exchange Digital Goods. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security (CCS)* (2018).

[26] Eberhardt, J., and Tai, S. ZoKrates - Scalable Privacy-Preserving Off-Chain Computations. In *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (2018).

[27] Fang, M., Cao, X., Jia, J., and Gong, N. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *29th USENIX Security Symposium (USENIX Security 20)* (2020).

[28] Fang, P., Zou, Z., Xiao, X., and Liu, Z. iSyn: Semi-automated Smart Contract Synthesis from Legal Financial Agreements. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)* (2023).

[29] Feng, B., Qin, L., Zhang, Z., Ding, Y., and Chu, S. ZEN: An Optimizing Compiler for Verifiable, Zero-knowledge Neural Network Inferences. In *Cryptology ePrint Archive* (2021).

[30] Fernandez, R. C., Subramaniam, P., and Franklin, M. J. Data Market Platforms: Trading Data Assets to Solve Data Problems. In *Proceedings of the VLDB Endowment* (2020).

[31] Fung, C., Yoon, C. J., and Beschastnikh, I. The Limitations of Federated Learning in Sybil Settings. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)* (2020).

[32] Garay, J., Kiayias, A., and Leonardos, N. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference (CRYPTO)* (2017).

[33] Ghorbani, A., and Zou, J. Data Shapley: Equitable Valuation of Data for Machine Learning. In *International conference on machine learning* (2019).

[34] Goldwasser, S., Micali, S., and Rackoff, C. The Knowledge Complexity of Interactive Proof-systems. In *SIAM Journal on Computing (SICOMP)* (1989).

[35] Gorbunov, S. Algorand Releases First Open-Source Code of Verifiable Random Function. https://algorand.com/resources/algorand-announcements/algorand-releases-first-open-source-code-of-verifiable-random-function, 2018.

[36] Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., and Schofnegger, M. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *30th USENIX Security Symposium (USENIX Security 21)* (2021).

[37] Groth, J. On the Size of Pairing-based Non-interactive Arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)* (2016).

[38] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[39] Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. Towards Efficient Data Valuation Based on the Shapley Value. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019).

[40] Karimireddy, S. P., He, L., and Jaggi, M. Learning from History for Byzantine Robust Optimization. In *International Conference on Machine Learning* (2021).

[41] Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. Hawk: The Blockchain Model of Cryptography and Privacy-preserving Smart Contracts. In *IEEE Symposium on Security and Privacy (SP)* (2016).

[42] Kothapalli, A., Setty, S., and Tzialla, I. Nova: Recursive Zero-Knowledge Arguments from Folding Schemes. In *Advances in Cryptology – CRYPTO 2022: 42nd Annual International Cryptology Conference (CRYPTO)* (2022).

[43] Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., and Suciu, D. Toward Practical Query Pricing with Querymarket. In *ACM SIGMOD* (2013).

[44] Koutsos, V., Papadopoulos, D., Chatzopoulos, D., Tarkoma, S., and Hui, P. Agora: A Privacy-aware Data Marketplace. In *IEEE 40th International Conference on Distributed Computing Systems (ICDCS)* (2020).

[45] Krishnamachari, B., Power, J., Kim, S. H., and Shahabi, C. I3: An IoT Marketplace for Smart Communities. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)* (2018).

[46] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Tech. rep., University of Toronto, 2009.

[47] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. In *Neural computation* (1989).

[48] Li, P., Wang, Y., Liu, Z., Xu, K., Wang, Q., Shen, C., and Li, Q. Verifying the Quality of Outsourced Training on Clouds. In *European Symposium on Research in Computer Security* (2022).

[49] Li, Q., Liu, Z., and Xu, K. martFL: Enabling Utility-Driven Data Marketplace with a Robust and Verifiable Federated Learning Architecture. *arXiv preprint arXiv:2309.01098* (2023).

[50] Li, X., and Roth, D. Learning Question Classifiers. In *19th International Conference on Computational Linguistics (COLING)* (2002).

[51] Liang, J., Li, S., Jiang, W., Cao, B., and He, C. OmniLytics: A Blockchain-based Secure Data Market for Decentralized Machine Learning. In *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML (FL-ICML)* (2021).

[52] Lin, J., Du, M., and Liu, J. Free-riders in Federated Learning: Attacks and Defenses. In *arXiv preprint arXiv:1911.12560* (2019).

[53] Liu, Z., Xiang, Y., Shi, J., Gao, P., Wang, H., Xiao, X., Wen, B., and Hu, Y.-C. Hyperservice: Interoperability and Programmability across Heterogeneous Blockchains. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (CCS)* (2019).

[54] Liu, Z., Xiang, Y., Shi, J., Gao, P., Wang, H., Xiao, X., Wen, B., Li, Q., and Hu, Y.-C. Make Web3. 0 Connected. *IEEE Transactions on Dependable and Secure Computing (TDSC)* (2022).

[55] Lloyd, S. Least Squares Quantization in PCM. In *IEEE Transactions on Information Theory* (1982).

[56] Lyu, L., Xu, X., Wang, Q., and Yu, H. Collaborative Fairness in Federated Learning. In *Federated Learning: Privacy and Incentive* (2020).

[57] Lyu, L., Yu, J., Nandakumar, K., Li, Y., Ma, X., Jin, J., Yu, H., and Ng, K. S. Towards Fair and Privacy-preserving Federated Deep Models. In *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2020).

[58] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient Learning of Deep Networks From Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)* (2017).

[59] Micali, S., Rabin, M., and Vadhan, S. Verifiable Random Functions. In *Annual Symposium on Foundations of Computer Science (FOCS)* (1999).

[60] Niu, C., Zheng, Z., Wu, F., Gao, X., and Chen, G. Achieving Data Truthfulness and Privacy Preservation in Data Markets. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2018).

[61] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)* (2019).

[62] Rabin, M. O. Transaction Protection by Beacons. In *Journal of Computer and System Sciences (JCSS)* (1983).

[63] Rousseeuw, P. J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. In *Journal of Computational and Applied Mathematics* (1987).

[64] Thorndike, R. Who Belongs in the Family? In *Psychometrika* (1953).

[65] Tibshirani, R., Walther, G., and Hastie, T. Estimating the Number of Clusters in a Data Set Via the Gap Statistic. In *Journal of the Royal Statistical Society Series B: Statistical Methodology* (2001).

[66] Voigt, P., and Von dem Bussche, A. The EU General Data Protection Regulation (GDPR): A Practical Guide.

[67] Wahby, R. S., Tzialla, I., Shelat, A., Thaler, J., and Walfish, M. Doubly-Efficient zkSNARKs Without Trusted Setup. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018).

[68] Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. In *Ethereum Project Yellow Paper* (2014).

[69] Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. In *arXiv preprint arXiv:1708.07747* (2017).

[70] Xie, T., Zhang, J., Cheng, Z., Zhang, F., Zhang, Y., Jia, Y., Boneh, D., and Song, D. zkBridge: Trustless Cross-chain Bridges Made Practical. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2022).

[71] Xu, X., and Lyu, L. A Reputation Mechanism Is All You Need: Collaborative Fairness and Adversarial Robustness in Federated Learning. In *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML (FL-ICML)* (2021).

[72] Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning (ICML)* (2018).

[73] Yoon, K. Convolutional Neural Networks for Sentence Classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014).

[74] Zamyatin, A., Harz, D., Lind, J., Panayiotou, P., Gervais, A., and Knottenbelt, W. Xclaim: Trustless, Interoperable, Cryptocurrency-Backed Assets. In *IEEE Symposium on Security and Privacy (SP)* (2019).

[75] Zhang, X., Zhao, J., and LeCun, Y. Character-Level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)* (2015).

[76] Zhu, Z., Shu, J., Zou, X., and Jia, X. Advanced Free-rider Attacks in Federated Learning. In *the 1st NeurIPS Workshop on New Frontiers in Federated Learning Privacy, Fairness, Robustness, Personalization and Data Ownership* (2021).