BlockFLow: An Accountable and Privacy-Preserving Solution for Federated Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Federated learning enables the development of a machine learning model among collaborating agents without requiring them to share their underlying data. However, malicious agents who train on random data, or worse, on datasets with the result classes inverted, can weaken the combined model. BlockFLow is an accountable federated learning system that is fully decentralized and privacy-preserving. Its primary goal is to reward agents proportional to the quality of their contribution while protecting the privacy of the underlying datasets and being resilient to malicious adversaries. Specifically, BlockFLow incorporates differential privacy, introduces a novel auditing mechanism for model contribution, and uses Ethereum smart contracts to incentivize good behavior. Unlike existing auditing and accountability methods for federated learning systems, our system does not require a centralized test dataset, sharing of datasets between the agents, or one or more trusted auditors; it is fully decentralized and resilient up to a 50% collusion attack in a malicious trust model. When run on the public Ethereum blockchain, BlockFLow uses the results from the audit to reward parties with cryptocurrency based on the quality of their contribution. We evaluated BlockFLow on two datasets that offer classification tasks solvable via logistic regression models. Our results show that the resultant auditing scores reflect the quality of the honest agents' datasets. Moreover, the scores from dishonest agents are statistically lower than those from the honest agents. These results, along with the reasonable blockchain costs, demonstrate the effectiveness of BlockFLow as an accountable federated learning system.

1 Introduction

Machine learning models benefit from large, diverse training datasets. For example, in healthcare, Electronic Health Records (EHR) provide a rich dataset for machine learning models that predict diseases [43]. Though this would greatly benefit society at large, the sensitivity of the data and strict government regulations such as HIPAA [2] restrict how hospitals can share such data with other entities. As such, entities with sensitive datasets can only develop locally-optimal models. However, for robust, globally-optimal models, that are highly generalizable, training needs to happen across organizational boundaries. Federated learning facilitates this by enabling the development of global models without sharing sensitive data [26]. Individual agents train local models on their dataset and share their local models with a centralized agent, which computes and shares the averaged model. This process repeats for multiple rounds until convergence. [30] demonstrated how federated learning leads to models that are stronger than any individual participant's local model. However, naive federated learning implementations are susceptible to several privacy and accountability threats.

In an N agent federated learning setup, $N-1$ colluding agents can compute the remaining agent's model. With the raw model, they can perform model inversion [19, 40] and membership inference attacks [32, 33, 36]. Differential privacy [15], the process of adding random noise to the model parameters [6, 8, 31], prevents these attacks. However, differential privacy does not protect against malicious agents that do not follow the protocol. For example, consider a logistic regression experiment where some agents invert their output labels, train, and submit a model based on this data. Such adversarial models can worsen the shared model [24]. Other agents would then use this worsened shared model in the next round, resulting in reduced overall accuracy [29, 41]. To defend against such attacks, agents must be held *accountable* for their contributions. Evaluating all contributions can prevent malicious contributions from being included in the shared model. Moreover rich evaluation can reward good contributors. Though it is trivial for a centralized agent to evaluate models such as by measuring accuracy on a secret (test) dataset, identifying a trusted, centralized agent is not always feasible (e.g. in international collaborations). Thus, a decentralized federated learning process, which is resilient to a minority of malicious agents, is ideal.

In addition, we propose the use of decentralized computational platforms, such as blockchains, for accountability. Blockchains combine an immutable ledger, currency, and with the Ethereum blockchain, a Turing-complete computational environment [39]. Programs that run in a blockchain's computational environment are known as "smart contracts." All blockchain data is public, and the Ethereum consensus protocol ensures that smart contracts will be executed correctly. Thus, agents can trust that any data sent to the ledger will not be tampered with and that any blockchain currency ("cryptocurrency") sent to smart contracts will only be released according to the contract's logic. With these guarantees, blockchains are appropriate to run federated learning accountability procedures and redistribute funds based on the results of these procedures.

BlockFLow is an accountable federated learning system that is fully decentralized and privacy-preserving. Its primary goal is to reward agents in relation to the quality of their individual contributions while protecting the privacy of the underlying datasets and being resilient to malicious adversaries. To accomplish these goals, BlockFLow requires no central trusted agent, enables agents to share differentially private models with each other and uses blockchain smart contracts to score each agent's contribution and provide monetary incentives for good behavior. Unlike existing auditing and accountability methods for federated learning systems, our system does not require a centralized test dataset, sharing of datasets between the agents, or one or more trusted auditors [3, 25].

Our paper is organized as follows: In section 2, we analyze other approaches to secure federated learning systems. Section 3 details the BlockFLow system design. Section 4 elucidates BlockFLow's resistance against attacks. In section 5, we present experiments that validate BlockFLow's theoretical properties. In section 6, we summarize our contributions and discuss future work. We conclude with a discussion of the overall broader impacts of our work.

2 Background and Related Work

Given the wide attack surface for federated learning, numerous techniques have been proposed to mitigate privacy and accountability threats. However, unlike our work, previous approaches use a semi-honest trust model, require trusted agent(s), expose datasets, or do not defend against individual malicious agents.

2.1 Differential Privacy

Differential privacy [16] is a mathematical framework that measures the risk of including an individual in a dataset. Differentially private methods can learn statistics of a dataset as a whole without learning sensitive information of individuals in the group. It guarantees with high probability that the released results do not reveal which individuals are in the underlying dataset. Formally, a randomized mechanism $\mathcal{M}$ satisfies ϵ-differential privacy when there exists $\epsilon > 0$, such that $\Pr\left[\mathcal{M}(D_1) \in S\right] \leq e^\epsilon \Pr\left[\mathcal{M}(D_2) \in S\right]$ holds for every $S \subseteq \text{Range}(\mathcal{M})$ and for all datasets D_1 and D_2 differing on at most one element.

For any real-valued query function $t : \mathcal{D} \to \mathbb{R}$, where $\mathcal{D}$ denotes the set of all possible datasets, the global sensitivity Δ of t is defined as $\Delta = \max_{\mathcal{D}_1 \sim \mathcal{D}_2} |t(\mathcal{D}_1) - t(\mathcal{D}_2)|$, for all $\mathcal{D}_1 \in \mathcal{D}$ and $\mathcal{D}_2 \in \mathcal{D}$.

The Laplacian Mechanism preserves ϵ-differential privacy [15]. Given Δ of the query function t, and the privacy loss parameter ϵ, *Laplacian mechanism* $\mathcal{M}$ uses random noise X drawn from the symmetric Laplace distribution with scale $\lambda = \frac{\Delta}{\epsilon}$. For differentially private logistic regression, [9] proposed $\Delta = \frac{2}{d * \alpha}$, where d is the size of the participant's dataset and α is the logistic regularization parameter. Thus, adding noise drawn from $Laplacian(0, \frac{2}{d\alpha\epsilon})$ to logistic regression weights guarantees ϵ-differential privacy.

Using differential privacy in federated learning can prevent collusion-based attacks and model inference attacks [20]. BlockFLow uses differential privacy to preserve the privacy of the agents participating in the federated learning system.

2.2 Multi-Party Computation

Multi-party computation (MPC) [4, 10, 21, 35, 42] guarantees that agents do not learn anything other than the final output. An MPC-based averaging mechanism can be used in federated learning for privacy, however, it is susceptible membership inference attacks and model inversion attacks. Additionally, MPC requires a semi-honest threat model, which assumes that all agents follow the protocol correctly. BlockFLow offers a malicious agent threat model, where adversarial agents can be present, and uses differential privacy that prevents membership inference and model inversion attacks and thus provides stronger guarantees than MPC-based techniques.

2.3 Blockchain-Based Accountability for Federated Learning

Blockchain platforms have been used previously for accountable federated learning. [22, 27] propose systems for evaluating models directly on the blockchain, via an initially-hidden but verifiable test dataset. While trust-less, these proposals require datasets to be publicly revealed on the blockchain and incur significant blockchain computational expenses for on-chain model evaluation. These proposals are only applicable for small models with public datasets.

Instead of a public on-chain evaluation, [3, 25] propose using trusted aggregators and evaluators. These systems assume that all agents agree to trust centralized agent(s). While these proposals offer better privacy guarantees, for no data is publicly revealed, their weak trust model significantly limits their applicability.

Finally, [11] proposed a robust, trust-less, gradient-based validation scheme, where only differentially private gradient updates are revealed on a public blockchain. It only includes the gradient updates most similar to the average update. While both accountable and privacy preserving, error increases proportionally to the number of malicious agents. It also does not scale well to large models, for gradients are averaged on the public blockchain.

BlockFLow offers numerous advantages compared to these existing works. First, it is both privacy-preserving and trust-less: it does not require any secret test dataset, trusted set of agents, or the revealing of data or weights on a public blockchain. It also supports a robust threat model. So long as there are a minority of malicious agents, BlockFLow filters out all malicious agents and thus preserves shared model quality.

3 Proposed System

BlockFLow is an accountable federated learning system that is fully decentralized and privacy-preserving. It leverages differential privacy to protect individual agents' datasets and an Ethereum blockchain smart contract to provide accountability. We provide reference implementations in the supplementary materials (appendix B) for both the Ethereum smart contract and the client code that interacts with the smart contract.

3.1 BlockFLow Client

Each agent must run their own instance of the BlockFLow client, which handles all aspects of an agent's participation in a federated learning experiment. For every federated learning round, each BlockFLow client trains a local model, applies differential privacy by adding Laplacian noise to the model, shares its model with the other clients in the experiment, retrieves and evaluates other clients'

models, reports the evaluation scores to the BlockFLow smart contract, and retrieves the overall
scores and averages the clients' models.

The BlockFLow client interacts with the BlockFLow smart contract through an Ethereum client,
which is required to communicate with the Ethereum network. Agents can run their own Ethereum
client, such as Geth [18], or use a hosted Ethereum client, such as Infura [23].

Since storing data on the Ethereum blockchain is expensive [34], BlockFLow uses the InterPlanetary
File System (IPFS) to share the differentially private models during each federated learning round.
IPFS is peer to peer, and anyone can host any content. IPFS provides automatic tamper detection
via cryptographic hashing [5]. Before uploading models to IPFS, clients encrypt their models using
Elliptic Curve Diffie Hellman [38] keys derived from the sender's and receiver's Ethereum accounts.
Since IPFS is publicly accessible (and does not have any built-in authentication), encryption prevents
external agents from accessing clients' models.

A detailed description of the client procedure is included in appendix A.

3.2 Ethereum Smart Contract

The BlockFLow Ethereum smart contract implements our novel contribution scoring procedure.
Each BlockFLow experiment must deploy its own instance of the BlockFLow smart contract on the
Ethereum blockchain. In order to initialize and deploy the smart contract, the following needs to be
specified: the number of federated learning rounds, a list of Ethereum accounts who are permitted to
join the experiment, bond parameters to control the refund rewarding, and timing parameters that
control the duration for each stage in the experiment.

At the beginning of an experiment, the smart contract requires that every client posts a bond (in Ether,
the Ethereum cryptocurrency) to join the experiment. This bond is used to ensure that clients follow
the protocol and to reward or penalize clients for their contributions.

Each federated learning round consists of five stages: `train`, `retrieve`, `evaluation commit`,
`evaluation reveal`, and `compute score`. The smart contract enforces strict deadlines for each
stage; clients who miss a deadline are eliminated from the smart contract and lose their posted bond.

Before the `train` deadline, clients must submit the IPFS address for their trained model to the smart
contract. Before the `retrieve` deadline, each client reports to the smart contract the set of clients
whose models were successfully validated. Next, the smart contract computes which clients were
able to retrieve both the majority of the other clients' models and have other clients retrieve their
model. Clients who fail this test are eliminated. Clients are required to evaluate all agents who pass
this test. Before the `evaluation submit` deadline, clients must submit evaluation scores encrypted
with unique salts.[1] Then, before the `evaluation reveal` deadline, all clients must reveal their
salts and decrypted evaluation scores to the smart contract. The smart contract verifies the salts
and scores. Finally, in the `compute score` stage, the smart contract then executes the contribution
scoring procedure as specified in section 3.3 to determine the overall scores for each client. The smart
contract then uses these scores to perform a *pro rata* partial refund of the bond among the clients, and
clients use these scores to determine how heavily each model has to be weighted while averaging.

This process is repeated for every federated learning round. After the final round, the remaining bond
is refunded among the clients.

3.3 Contribution Scoring Procedure

BlockFLow requires each client to evaluate every other client's model during each federated learning
round. Clients are encouraged to use their entire dataset to perform evaluations. For the set of models
W and evaluation function `eval` the following properties must hold:

$$(\forall w, w' \in \{W \times W\}, w \neq w') \quad \texttt{eval}(w) > \texttt{eval}(w') \iff \text{model } w \text{ is better than model } w'$$
$$(\forall w \in W) \quad \texttt{eval}(w) \in [0.0; 1.0]$$

For classification problems, F1 or accuracy scores satisfy the above properties. The clients must
agree on which formula to use.

---

[1]A salt is random data used as an additional input to a one-way hash function to prevent dictionary attacks.

4

Clients' overall scores reflect the minimum of a) the median score reported for their model (as determined by all clients) and b) the inverse of the maximum difference between one's reported score and the median score for each model. This procedure incentivizes clients to share high-quality models and perform high-quality evaluations. Submitting a low-quality model would result in a low model median score, and submitting inaccurate evaluations would result in a significant maximum difference from the median. Algorithm 1 provides a formal description of the scoring procedure.

Algorithm 1: Contribution Scoring Procedure. This procedure, except for the `eval` method, is implemented in the blockchain smart contract.

---

for all client pairs $\{a, k\} \in \{N \times N\}$ **do**
    $s_{a,k} \leftarrow \mathtt{eval}_a(k)$ {client a performs off-chain evaluation of client k's model using a's own dataset and reports the score here. See section 3.3 for `eval` requirements.}
end for
for all clients $k \in N$ **do**
    $m_k \leftarrow \mathrm{MED}\{s_{a,k} : \forall a \in N\}$ {Compute the median model scores}
end for
for all clients $k \in N$ **do**
    $m'_k \leftarrow \frac{m_k}{\max\{m'_k : \forall k' \in N\}}$ {Scale the median model scores to ensure a maximum of 1.0}
end for
for all client pairs $\{a, k\} \in \{N \times N\}$ **do**
    $t_{a,k} \leftarrow |s_{a,k} - m_k|$ {Compute the evaluation quality scores}
    $t'_{a,k} \leftarrow \max\left(0, \frac{0.5 - t_{a,k}}{0.5 + t_{a,k}}\right)$ {Transform the evaluation quality scores}
end for
for all clients $a \in N$ **do**
    $d_a \leftarrow \min\{t'_{a,k} : \forall k \in N\}$ {Compute the least accurate evaluation each client performed}
end for
for all clients $a \in N$ **do**
    $d'_a \leftarrow \frac{d_a}{\max\{d_{a'} : \forall a' \in N\}}$ {Scale the least accurate evaluation scores}
end for
for all clients $k \in N$ **do**
    $p_k \leftarrow \min(m'_k, d'_k)$ {Compute the overall scores}
end for
return p

---

4 Threat Analysis

BlockFLow offers a 50% malicious threat model. Specifically, in an experiment with N agents, BlockFLow is resistant up to $M \in [0, \frac{N}{2})$ agents neglecting to follow the BlockFLow protocol for the experiment to maintain its integrity. We explore all known attacks on the BlockFLow system, on all components of the BlockFLow system, and offer how BlockFLow is resistant to such attacks.

4.1 Privacy Threat Model

By the definition of differential privacy, with extremely high probability, $N-1$ colluding agents cannot infer any information about the remaining agent [7]. Note that $(\forall N \geq 3) \quad N - 1 > \frac{N}{2}$.

4.2 Ethereum Blockchain Threat Model

Public/private key cryptography and a proof-of-work consensus protocol secure the Ethereum Blockchain [39]. There are no feasible attacks on the Ethereum Network, without controlling 50% of the computational power of the *entire* Ethereum network. Such an attack is estimated to cost nearly $150,000 USD per hour [13] and has never been successful on the Ethereum mainnet [12]. As such, we do not consider it to be a feasible threat to BlockFLow.

As the Ethereum Blockchain is public and anonymous, clients could theoretically enroll multiple times in an experiment and thus have a disproportionate vote. This is analogous to ballot box stuffing [28].

BlockFLow mitigates this attack through a restricted enrollment model, where only pre-approved Ethereum accounts can join an experiment. Through decentralized identity verification [1, 37] or manual processes, agents can ensure that each other agent controls only one account. Such verification is beyond the scope of BlockFLow. So long as there are a majority of honest agents with only one enrolled account (and thus, one vote), malicious (colluding) agents could never control 50% or more of the accounts in a BlockFLow experiment.

4.3 IPFS and Data Sharing Threat Model

IPFS is immutable [5] and agents cannot change their model after submitting the cryptographic hash to the smart contract. The BlockFLow protocol requires each agent to report if it is able to load strictly more than $\frac{N}{2}$ models, and have strictly more than $\frac{N}{2}$ agents report the same for one's own model. The BlockFLow threat model guarantees that there are strictly more than $\frac{N}{2}$ honest agents, meaning that all honest agents alone would fulfill these conditions for each-other.

This scheme also guarantees that for all agents a who meet these conditions, at least one different and honest agent $a' \neq a$ retrieved a's model. In the worst-case scenario, of the strictly more than half of the agents who retrieved a's model, strictly less than half are dishonest; the difference between these sets is never empty. Since IPFS allows anyone to share any content, one or more honest parties would share the model with all other agents. Thus, agents who are unable to retrieve a model directly from the source (e.g. due to firewall restrictions) would still be able to obtain all necessary models. Hence, this model sharing scheme satisfies the BlockFLow threat model.

4.4 Contribution Scoring Procedure Threat Model

There are several attacks possible on the contribution scoring procedure. First, we define *malicious models* to have weights that are not reflective of a truthful dataset. For example, models trained on randomly generated data or on inverted output features are considered malicious. Naively averaging such models into a global model would likely harm the shared objective.

The BlockFLow contribution scoring procedure penalizes those who submit malicious models. All agents evaluate every other agent's model. The median score an agent's model receives handicaps that agent's overall score; lower scores result in less cryptocurrency received. In addition, when computing the global model at the end of each round, agents are encouraged to weight models in proportion to the overall scores received.

Next, we consider agents who collude during the training process to submit better models. For example, agents can secretly share raw data or models among $M < \frac{N}{2}$ colluding agents. All M agents would then submit their best model. BlockFLow rewards agents who contribute strong models, and it is acceptable for multiple agents to submit identical models. All submitted models are evaluated independently and relative to each other. As such collusion is no different from having many agents with strong datasets, we do not consider training collusion to be an attack.

Finally, we explore attacks during evaluation. Through encryption and a commit-then-reveal protocol, the BlockFLow smart contract prevents agents from copying others' scores without collusion. Also, the Contribution Scoring Procedure maps any difference greater than 0.5 away from the model's median to a 0 score, as the a priori score is 0.5. Thus, agents are encouraged to evaluate honestly.

With collusion, a group of dishonest agents could submit the same realistic score, or even fabricate scores. Consider the extreme case where a minority subset of malicious agents report perfect 1.0 scores for their models and 0.0 scores for all others (e.g. models from honest agents). However, since there are strictly less than half malicious agents, and only the median model scores are used to determine one's overall score, the median score is guaranteed to be between the minimum and maximum scores reported by the honest agents. As any honest agent's score is a fair evaluation, it is impossible for colluding agents to materially affect the evaluation scores.

Moreover, the fabrication of scores will only penalize those who attempt it. The contribution scoring procedure depicted in algorithm 1 limits one's overall score with the evaluation on which one was furthest away from the median. Specifically, any agent with an evaluation of more than 0.5 away from the median will receive an overall score of 0, and no share of the cryptocurrency pool. So long as the median scores are bounded by the range of the honest agents' scores, which our threat model guarantees, fabrication is never optimal.

5 Evaluation

We evaluated our system on the Adult Census Income (Adult) [14] and The Third International Knowledge Discovery and Data Mining Tools Competition (KDD) [14, 17] classification tasks. All discrete features are one-hot encoded, and continuous features were scaled globally and independently to be between 0 and 1. Two-thirds of the data was split in equal shares (unless otherwise noted) among the N clients, without overlap, and clients reserved 20% of their data for training validation. Clients used their entire datasets when evaluating others' models. The remaining one-third was reserved for testing. All experiments used logistic regression models with L2 regularization coefficient $\alpha = 1.0$ and F1 scores as the `eval` function.
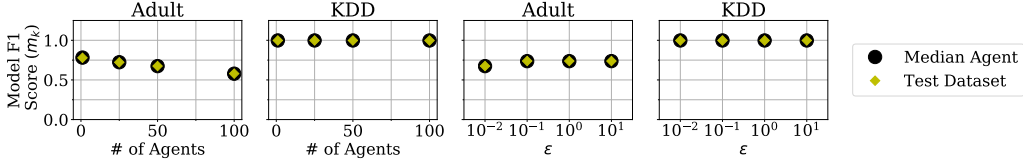

Figure 1: **Experiment 1: Measuring Model Quality** The above charts show the average of median agent F1 scores (m_k from algorithm 1) and average of reserved test dataset F1 scores. $\epsilon = 0.01$ when varying $N \in \{1, 25, 50, 100 \text{ agents}\}$, and $N = 50$ agents when varying the privacy parameter $\epsilon \in \{0.01, 0.1, 1, 10\}$. Across all evaluations, the average absolute difference is $< 0.67\%$.

In the first experiment, we validated that the median of the agents' F1 scores align with the one-third test dataset F1 scores, which we consider to represent absolute model quality. Agents did not train or evaluate on such test datasets. Figure 1 illustrates the similarity between these scores for all evaluated (N, ϵ) configurations. Thus, the median F1 score, even when individual agents have relatively small datasets, accurately estimates model quality.

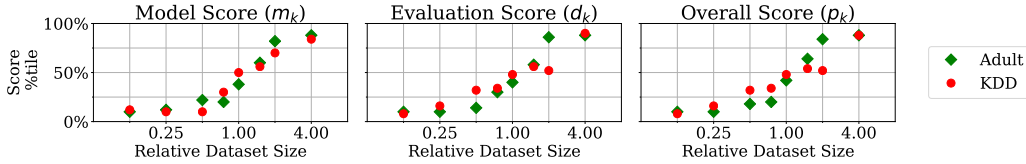Figure 2: **Experiment 2: Rewarding Larger Datasets** Agent's model, evaluation, and overall scores (m_k, d_k, and respectively p_k from algorithm 1). Of $N = 50$ agents, we varied the relative dataset size for 10 agents. $\epsilon = 0.01$. For both datasets and all three scores, correlation $\rho(\log(\text{relative dataset size}), \text{score percentile}) > 0.9$. Hence, it is possible to identify and reward those with large datasets while maintaining dataset privacy and without sharing underlying data.

In the second experiment, we evaluated whether the contribution scoring procedure can reward those with higher-quality datasets, as higher-quality datasets should result in robust models and accurate evaluations. As shown in figure 2, agents' model, evaluation, and overall scores were strongly correlated with their dataset size ($\rho > 0.9$ for all evaluations). As such, this scoring procedure can identify and reward (via cryptocurrency) those with larger datasets, without ever sharing the underlying data.

In the third experiment, we examined evaluation collusion attacks. Specifically, consider the fabrication of evaluation scores where $M < \frac{N}{2}$ clients award perfect (1.0) scores to other colluding clients. Figure 3 illustrates how such collusion is reflected in the colluding agents' statistically lower evaluation and overall scores.

In the fourth experiment, we explored malicious training setups, where a minority subset $M < \frac{N}{2}$ of the agents submit models trained on *random* or *inverted* data. Independently sampling each feature with probabilities from the real dataset distribution formed datasets for random agents. Flipping the output labels created the inverted agents' datasets. Like honest clients, malicious clients split their
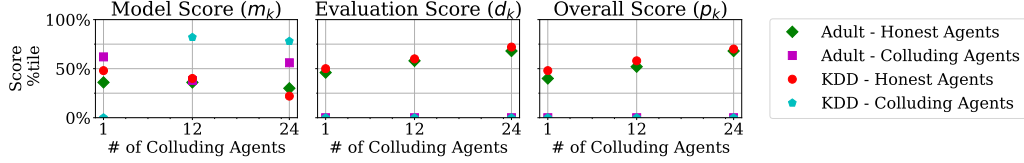
Figure 3: **Experiment 3: Discouraging Collusion** Average colluding and non-colluding agents' model, evaluation, and overall scores (m_k, d_k, and respectively p_k from algorithm 1). $N = 50$ agents, $\epsilon = 0.01$. Agents colluded by rewarding other colluding agents perfect scores. While it appears that colluding agents obtained better model scores, note the statistically lower ($p < 10^{-31}$) overall scores for colluding agents compared to the honest agents. Hence, it is a sub-optimal strategy to fabricate evaluation scores.
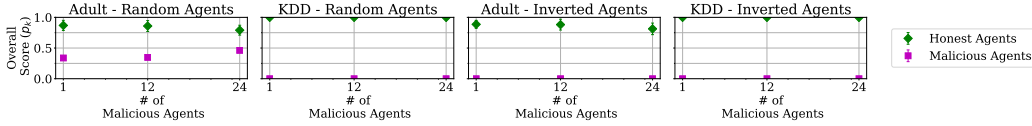

Figure 4: **Experiment 4: Punishing Malicious Agents** Average honest and malicious agent's overall score (p_k from algorithm 1). $N = 50$ agents, $\epsilon = 0.01$. Random agents trained on random data; inverted agents trained out inverted output features. For all $N > 1$ agent experiments, malicious agents' scores were statistically lower ($p < 10^{-22}$).

datasets into 80% train and 20% validate portions. Figure 4 illustrates how the malicious agents scores were statistically lower than those of the honest agents.

Finally, we present an analysis of the blockchain costs of our system. Ethereum measures computational costs in terms of gas, which incorporates the amount of storage used and CPU instructions executed [39]. While the amount of gas is independent of market conditions, such as the price of gas or the exchange rate of Ethereum, it is directly proportional to real world costs. A linear regression ($R^2 > 0.99$) determined that per-agent gas consumption for N agents and R federated learning rounds is

$$\text{gas}(N, R) = 31913N + 542045R + 477050NR$$

Such costs could be lowered by preferring off-chain computation of algorithm 1 and using the on-chain implementation only if any agent disagrees with the off-chain calculated scores.

Collectively, these experiments illustrate the robustness of the contribution scoring procedure and validate its theoretical properties discussed in section 4.4. The Laplacian Differential Privacy Method [15], Ethereum Blockchain [39], and IPFS [5] and data sharing protocol are themselves resilient up to at least a 50% attack. We have shown that the contribution scoring procedure is equally secure. As such, the entire BlockFLow system is resilient up to a 50% malicious agent attack.

6   Conclusion and Future Work

By offering an accountable federated learning system that is fully decentralized and privacy-preserving, BlockFLow closes the trust and accountability gap with federated learning. No existing system simultaneously offers accountability guarantees, privacy preservation, and a fully trust-less, malicious agent threat model. This combination of features expands the reach for federated learning research, for example to international collaborations where agents need not know nor trust each-other but are rewarded for contributing high quality models.

BlockFLow can be extended to improve its performance. Specifically, its asymptotic costs, for N agents and R federated learning rounds, scale with $O(N^2R)$, since for each round, each agent must evaluate all other agents' models. Instead of requiring all other agents' contributions, the smart contract could randomly select $Q << N$ agents to evaluate each agent's work. A sufficiently large Q, with high probability, would lead to accurate results and be resistant to $M < \frac{N}{2}$ malicious agents. This trade-off would greatly reduce gas consumption and increase scalability in large experiments.

Broader Impact

This research is aimed at an often neglected aspect in federated machine learning - accountability of agents. Accountability is built into society through regulations and social norms but is rarely reflected in our technologies. Our research models how social accountability works through punishments and rewards and uses blockchain technologies to enforce them monetarily. We believe that this will be extremely useful for banks, hospitals and other organizations who want train machine learning models collaboratively but deal with very sensitive data and where erroneous training data could have serious consequences. For example, someone may try to deliberately sabotage the model (e.g. training on diabetes patients when in fact they are heart patients, or labeling babies as adults). BlockFLow can co-ordinate such experiments by ensuring that individuals' private information is not shared, detecting agents who are contributing poor models, and being resistant to 50% collusion of malicious agents who try to sabotage the collaboration process.

Federated learning ensures that models do not overfit and are highly generalizable. For example, health-care models trained on data collected from different hospitals located in different continents generalize better. Also, it is important to collect data from all demographics to build a fair classifier. BlockFLow leads to more unbiased and fair AI because it enables more generalizable models to be easily developed via federated learning.

BLockFlow provides a payment mechanism to pay for high-quality (i.e. robust) models. Agents trying to cheat or induce bias will be caught with high probability, and the penalty cost imposed for cheating is more than the reward in expectation. This mechanism, along with BlockFLow's privacy guarantees and threat model, closes the trust and accountability gaps with federated learning. No existing system offers similarly strong properties. As such, BlockFLow enables new types of federated learning.

However, BlockFLow has some shortcomings that should be taken into consideration while designing the federation experiment. BlockFLow only compensates the data custodian, not the data owner. For example, patients are not directly compensated when a hospital uses patient data. BlockFLow cannot enforce that such compensation be made since differential privacy prevents anyone from knowing whether a particular individual was included in the training dataset.

In addition, agents who lack computational resources or lack quality data would not be able to train a strong model and henceforth would receive less compensation. Though their contribution was valuable, such agents could end up losing money as their contribution was limited. To address this, the scoring algorithm could be made binary (i.e.) all agents, whose models are of a certain quality, get the same reward rather than the better model getting a larger reward. In addition, agents with limited resources who cannot deposit a bond are prevented from participating in the federation system.

Lastly, if M malicious agents ($M \geq \frac{N}{2}$) participate in the system, the system could fail and could result in a sub-optimal model. Along with this, the honest agents participating in the federation could lose their monetary benefits as well.

References

[1] Anurag Angara, Andy Chorlian, Shane Hampton, and Noah Zinsmeister. Eip 1484: Digital identity aggregator. `https://eips.ethereum.org/EIPS/eip-1484`.

[2] George J Annas et al. Hipaa regulations-a new era of medical-record privacy? *New England Journal of Medicine*, 348(15):1486–1490, 2003.

[3] Sana Awan, Fengjun Li, Bo Luo, and Mei Liu. Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2561–2563, 2019.

[4] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[5] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[6] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

[7] Vincent Bindschaedler, Shantanu Rane, Alejandro E Brito, Vanishree Rao, and Ersin Uzun. Achieving differential privacy in secure multiparty data aggregation protocols on star networks. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 115–125. ACM, 2017.

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[9] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in neural information processing systems*, pages 289–296, 2009.

[10] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, page 462, 1987.

[11] Xuhui Chen, Jinlong Ji, Changqing Luo, Weixian Liao, and Pan Li. When machine learning meets blockchain: A decentralized, privacy-preserving and secure design. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1178–1187. IEEE, 2018.

[12] Coindesk. Coindesk-attack. `https://www.coindesk.com/tag/51-attack`.

[13] Tom Dickman. Crypto51. `https://www.crypto51.app/`.

[14] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[16] Cynthia Dwork and Adam Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010.

[17] Charles Elkan. Results of the kdd'99 classifier learning. *Acm Sigkdd Explorations Newsletter*, 1(2):63–64, 2000.

[18] Go Ethereum. Official go implementation of the ethereum protocol. *URl: https://geth. ethereum. org (visited on 01/25/2019)*, 2017.

[19] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[20] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[21] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[22] Justin D Harris and Bo Waggoner. Decentralized and collaborative ai on blockchain. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 368–375. IEEE, 2019.

[23] Infura. Secure and scalable access to ethereum apis and ipfs gateways. `https://infura.io/`.

[24] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[25] Hyunil Kim, Seung-Hyun Kim, Jung Yeon Hwang, and Changho Seo. Efficient privacy-preserving machine learning for blockchain network. *IEEE Access*, 7:136481–136495, 2019.

[26] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[27] A Besir Kurtulmus and Kenny Daniel. Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain. *arXiv preprint arXiv:1802.10185*, 2018.

[28] Fabrice E Lehoucq and Iván Molina. *Stuffing the ballot box: fraud, electoral reform, and democratization in Costa Rica*. Cambridge University Press, 2002.

[29] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1544–1551, 2019.

[30] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 3, 2017.

[31] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[32] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[33] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 739–753. IEEE, 2019.

[34] Paul Rimba, An Binh Tran, Ingo Weber, Mark Staples, Alexander Ponomarev, and Xiwei Xu. Comparing blockchain and cloud services for business process execution. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 257–260. IEEE, 2017.

[35] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[36] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[37] Fabian et al. Vogelsteller. erc725alliance. `https://erc725alliance.org/`.

[38] Shengbao Wang, Zhenfu Cao, Maurizio Adriano Strangio, and Lihua Wang. Cryptanalysis and improvement of an elliptic curve diffie-hellman key agreement protocol. *IEEE communications letters*, 12(2):149–151, 2008.

[39] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[40] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370. IEEE, 2016.

[41] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *arXiv preprint arXiv:1912.12716*, 2019.

[42] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

[43] Tao Zheng, Wei Xie, Liling Xu, Xiaoying He, Ya Zhang, Mingrong You, Gong Yang, and You Chen. A machine learning-based framework to identify type 2 diabetes through electronic health records. *International journal of medical informatics*, 97:120–127, 2017.

A BlockFLow Client Procedure

Algorithm 2: BlockFLow Client Procedure

Require: agent $i \in N$; i posted bond and enrolled in BlockFLow smart contract

  repeat
    for all agents $k \in N$ **do**
      if agent k requests $w_{k'}^r$ for agent k' and $w_{k'}^r$ is valid **then**
        encrypt $w_{k'}^r$ and share with agent k
      end if
    end for
  until forever, in background process
  for round $r \leftarrow 1$ to R **do**
    repeat
      $w_i^r \leftarrow w_i^r - \eta \nabla g(w_i^r)$
    until deadline for round r
    $w_i^r \leftarrow w_i^r + Lap(0, \frac{2}{d_i \alpha \epsilon})$
    $a_i^r \leftarrow$ encrypt and upload model w_i^r to IPFS and record the address
    record model IPFS address a_i^r in BlockFLow smart contract
    for all agents $k \in n$ **do**
      $a_k^r \leftarrow$ get agent k model's IPFS address from smart contract
      $w_k^r \leftarrow$ load and validate model a_k^r
      if w_k^r is valid **then**
        report w_k^r as valid in BlockFLow smart contract
      end if
    end for
    wait for data retrieval deadline
    for all agents $k \in N$ **do**
      $v_k^r \leftarrow$ count number of models agent k marked as valid
      $z_k^r \leftarrow$ count number of agents who marked agent k's model as valid
      if $v_k^r \leq \frac{N}{2}$ or $z_k^r \leq \frac{N}{2}$ **then**
        $N \leftarrow N \setminus k$
      else
        if agent i does not have valid model w_k^r **then**
          for all agents $k' \in N$ **do**
            $w_k^r \leftarrow$ request and retrieve model k from agent k' {See section 4.3 }
          end for
        end if
        $s_{i,k}^r \leftarrow$ evaluate$_i(w_k^r)$
        $b_{i,k}^r \leftarrow$ random encryption key
        $s_{i,k}'^r \leftarrow$ encrypt$_{b_{i,k}^r}(w_k^r)$
        report encrypted score $s_{i,k}'^r$ for agent k to smart contract
      end if
    end for
    wait for before encrypted score submission deadline
    for all agents $k \in N$ **do**
      provide decryption key $b_{i,k}^r$ to provably reveal score $s_{i,k}^r$ to smart contract
    end for
    wait for score decryption submission deadline
    $p \leftarrow$ get scores from smart contract {See Algorithm 1}
    $w_i^{r+1} \leftarrow \frac{\sum_{k=1}^{n}(w_k^r * p_k)}{n * \sum_{k=1}^{n}(p_k)}$ {all agents should now have identical w_i^{r+1}}
  end for

B Reference Implementation

The reference implementation is available in the included `.zip` file.

12