Blockchain Enabled AI Marketplace: The Price You Pay For Trust

Abstract:

There has been a considerable amount of interest in exploring blockchain technologies for enabling marketplaces of different kinds. In this work, we provide a blockchain implementation that enables an "AI marketplace": a platform where consumers and data providers can transact data and/or models and derive value. Preserving privacy and trust during these transactions is a paramount concern. As an enabling use case, we consider a transfer learning setting. In this setting, a consumer entity wants to acquire a large training set from different private data providers that matches a small validation dataset provided by the consumer. Data providers expect a fair value for their contribution and the consumer also wants to maximize its benefit. We implement a distributed protocol on a blockchain that provides guarantees on privacy and consumer's benefit. We also demonstrate that our blockchain implementation plays a crucial role in addressing the issue of fair value attribution and privacy in a trustable way.We consider three different designs for a blockchain implementation that trades off trust requirements on different entities and the overhead in terms of time taken for completion of the task. The first design provides no trust guarantees. The second one guarantees trust with respect to other participants if the platform is trustworthy. The third one guarantees complete trust with no requirements. Our experiments show that the performance in the second and third cases, with partial/complete trust guarantees, degrade by roughly 2× and 5× respectively, compared to the baseline with no trust guarantees.

1. Introduction

In recent times, the fields of Artificial Intelligence and Machine Learning have revolutionized several industrial domains, particularly those related to computer vision applications. This has become possible partly due to the breakthroughs in modern hardware technologies such as GPUs and partly due to the massive amounts of image data that are collected by the media giants such as Google or Facebook.

In cross-enterprise domains, such as health care, banking, insurance or travel, data is typically spread across multiple players who own and protect it. For instance, different hospitals record data belonging to patients from different demographic distributions, or customer insurance data is spread across multiple insurance companies. In such scenarios, collaboration between various organizations in AI or ML tasks leads to significant performance gains. However, due to the confidential nature of enterprise data, one must protect its privacy and ownership. In particular, direct sharing of data is not acceptable in such scenarios. These strict privacy requirements lead to other challenging issues such as, *trust* - ensure that no collaborating entity can cheat in the model building process, *fairness* - entities should be fairly rewarded based on their contributions, *auditability* various aspects of the overall AI process, including any potential breaches in trust and fairness, should be verifiable by an independent third party agent. Note that this kind of verifiability may be useful even in a non-collaborative setting with privacy constraints. In this work, we demonstrate that blockchain [30, 24, 4, 8, 19, 9, 6, 28] technology, which is poised to revolutionize the financial sector [26], is not only well suited but in fact indispensable in addressing the above issues in an AI/ML model generation task.

Blockchain is a distributed ledger that allows multiple entities to transact in a secure and immutable fashion. This immutable ledger enables tamper-free provenance tracking of a process such as the AI model training. One key concern in using blockchain arises from issues surrounding latency and throughput. Our main contribution is evaluation of three different implementations that provide a trade-off between the level of trust and system performance in the context of a machine learning task.

To demonstrate the design principles involved, our use case is a protocol designed to enable an AI Marketplace. First we define the basic setting of an AI Marketplace : a consumer arrives with a small dataset, referred to as a "validation" dataset, and wants to build a prediction model that performs well on this dataset. However, the model training process requires huge amount of data, that it must acquire from multiple private sources. Fundamentally, the AI Marketplace must address a *transfer learning problem*, where the distribution of data at different sources is considerably different from each other and even from the validation dataset. The Marketplace must facilitate transactions of data points from multiple sources towards the consumer's task by forming a training dataset that is close in some distance measure to the validation dataset. In the process, it must enable monetization of the data in a trusted, fair manner while preserving data ownership and privacy as much as possible. Consider the following scenario in the health care domain, as an example. Suppose the consumer is a newly established cancer hospital and the data sources are cancer institutions from different geographical locations across the globe. The goal of the new hospital is to construct ML models that, say, can predict early onset of some form of cancer. The quality of the model depends on the demography of its patients and therefore it is crucial to collect data that matches a small validation set

that is representative of the demography. The individual sources clearly have widely different demographic data. The goal of an AI Marketplace is to enable private collection of a dataset sampled from these sources that matches the demography of the new institute and in the process attribute fair value to different data sources.

In a companion work [1], we proposed a distributed data exchange protocol, mediated by an aggregator entity, that enables the above AI Marketplace scenario. The protocol has the following properties:

- With respect to communication *to* a specific data owner, all other individual data sources and the consumer have privacy protection in the form of differential privacy guarantees. This is accomplished by ensuring that all communications (between the aggregator and various data owners) take place using some random hashes of data points.
- Facilitate transfer learning in the sense that the aggregator entity acquires a summary training dataset that is closely matched to the consumer validation dataset in some statistical distance (our choice is Maximum Mean Discrepancy). This is done by an iterative algorithm that incrementally acquires data points (in some hashed form).
- The aggregator, at the worst, can only learn the pairwise Euclidean distances between the points in the training set collected.

In this paper, we demonstrate how implementing the above protocol on a blockchain can enable us to address the issues of trust, fairness, auditability and value attribution in the above transfer learning setting. We first outline the private protocol that involves exchanges of random hashes of data points between the aggregator and data owners. We just quote the results from the companion work that provides formal guarantees on privacy of the protocol. Then, we discuss how to address the question of value attribution, fairness and trust and how a blockchain implementation can address these issues in the context of the above protocol turning it into a marketplace effectively. We also provide results from a real blockchain implementation and discuss tradeoffs.

2. The Problem

Given K data owners with private datasets denoted by $D_1, D_2, \ldots D_K$. Here, $D_i \in \mathbb{R}^{m_i \times n}$ where m_i denotes the number of points and n denotes the dimension of each of the data points. We also consider a validation set $D_v \in \mathbb{R}^{m \times n}$ which is private to another entity called 'consumer'. The consumer entity wants to form a summary data set $D_s \subseteq \bigcup_i D_i$ and $|D_s| = p$ such that D_s is close to D_v in the MMD (Maximum Mean Discrepancy) statistical distance defined below. The goal is to train a machine learning model (complex enough) on the summary dataset that could perform well on the test distribution that is identical to the consumer's validation set. In this work, we focus on the collection of the summary dataset.

Definition: The sample MMD distance for finite datasets $D \in \mathbb{R}^{m_1 \times n}$ and $D' \in \mathbb{R}^{m_2 \times n}$ as follows:

$$\mathrm{MMD}^2(D, D') = \frac{1}{m_1^2} \sum_{x,x' \in D} k(x, x') - \sum_{x \in D, y \in D'} \frac{2k(x, y)}{m_1 m_2} + \frac{1}{m_2^2} \sum_{y,y' \in D'} k(y, y') \quad (1)$$

where $k(\cdot,\cdot)$ is a kernel function underlying an RKHS (Reproducing Kernel Hilbert Space) function space such that $k(x,y) = k(y,x)$ and $k(\cdot,\cdot)$ is positive definite. For this work, we will use a Gaussian RBF kernel $k(\mathbf{x},\mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|_2^2)$ with some constant $\gamma > 0$.

Our objective is to form a summary D_s of size p by collecting points from all data owners. We maximize the following normalized MMD objective [15] as described below. For fixed validation set D_v such that $|D_v| = m$ and the summary set D_s, the objective we seek to optimize is $J(D_s)$ given by:

$$J(D_s) = \sum_{i,j \in D_v} \frac{k(y_i, y_j)}{m^2} - \mathrm{MMD}^2(D_v, D_s)$$

We have shown that the function is submodular under some weak assumptions on the kernel function. Submodularity [18, 10, 16, 11] permits a greedy algorithm where the points with the best marginal gain can be added iteratively that maximize the submodular objective. The general idea is for an entity called aggregator to maintain the summary D_s. Ideally, every data source k can look at $J(D_s)$ and add the best point x_k such that $x_k = \mathrm{argmax}_{x \in D_k} J(D_s + x) - J(D_s)$. The key issue is that if D_s is exposed at every iteration, a given data source would be able to find out the data points contributed by others violating privacy. We summarize challenges when using any traditional greedy approach and we list some of them here.

1. Data privacy: The raw data summary D_s cannot be shared directly with data sources as this will result in ownership loss.
2. Fair value attribution: We should ensure that the platform is completely transparent in assigning value

[1] The companion work is under submission and tackles only the privacy aspects providing formal differential privacy guarantees. However, our current submission discusses the blockchain implementation including many other aspects apart from privacy.

[2] Although important from the protocol's point of view, a reader may choose to skip the detailed description and justification of using these two hashes.

based on the quality of data provided towards to the target goal.

3. Trusted data sharing: Data providers should not be allowed to cheat by providing wrong inputs in the model building process.

4. Auditable and tamper-free tracking: The entire process must be transparent and immutable in order to ensure trust and fairness.

3. Addressing the Privacy Issues: Our Protocol

In a companion paper, we have designed a distributed protocol that addresses the concern of privacy with negligible impact on the solution of the greedy algorithm. We describe our protocol and explain how it is designed to address privacy.

We adopt the following definition of differential privacy in our work. On a high level, it means that two datasets that differ in at most one point should not cause a differentially private algorithm to produce output whose distribution that is very different. We state the formal definition below:

Definition 1 *The output of a randomized algorithm* $\mathrm{A}(D)$ *is* (ϱ,δ) *differentially private with respect to the input dataset D if for any two neighboring datasets D,D′ that differs in one data point,*

$$P\left(\mathrm{A}(D) \in E\right) \leq e^{\varrho} P\left(\mathrm{A}(D') \in E\right) + \delta. \qquad (2)$$

for all events E that can be defined on the output space.

Each data source must be able to compute the marginal gain without the aggregator having to share the data points directly. At the very least, the aggregator must expose some random function of the current summary D_s that is differentially private with respect to D_s such that each data source can still compute the best marginal gain approximately. Now we describe the key idea behind our protocol that addresses the privacy problem.

Key idea: Our protocol is essentially "a tale of two hashes" where data sources use a hash function $\mathbf{h}_1$ to release data points while the aggregator uses another hash function $\mathbf{h}_2$ to suitably expose the current summary (of $\mathbf{h}_1$ hashes acquired). $\mathbf{h}_1$ is not known to the aggregator (the random seed is known only to the data sources) while $\mathbf{h}_2$ is known only to the aggregator. However, the computations make progress approximately as in the standard greedy algorithm.

Detailed description of the hashes[2]: Observe that the kernel function is non-linear with two data point arguments. It is difficult to evaluate a new point without sharing the existing summary datapoint due to the non-linearity. To solve this problem, every data source has access to a random hash function $\mathbf{h}_1$ and only hashes of these data points is shared with the aggregator. Inspired by random Fourier features method of Rahimi and Recht [21], we design $\mathbf{h}_1$ such that $\mathbf{h}_1(x)^T\mathbf{h}_1(y) \approx k(x,y)$.

The above hash function reduces a non linear kernel computation to a dot product computation resulting in separability. The aggregator always obtains $\mathbf{h}_1(x)$ from the data sources. It only exposes an aggregate of the hashes corresponding to points in the current summary D_s. This would let any data source compute the marginal gain. This still does not solve the issue of privacy. The aggregator cannot expose a linear combination of h_1 hashes as is. The reason is that since the random hash function is known among all data sources, observing subsequent releases any data source could potentially compute data points contributed by others.

To ensure the privacy of aggregator's release during any communication with data owners, we use another random hash function $\mathbf{h}_2$ such that for $\mathbf{g} = \mathbf{h}_2(\mathbf{h}_1(D))$, $\mathbf{g}^T\mathbf{h}_1(x) \approx \frac{\mathrm{P}}{|D^1|}\sum_{y\in D} k(x,y)$. Further $\mathbf{h}_2$ is chosen such that the release $\mathbf{g}$ is differentially private with respect to D. Description of the Protocol: Now, we describe the protocol in some detail in Algorithm 1 to make this paper selfsufficient.

Initialization: The protocol is initiated with the consumer using the hash function $\mathbf{h}_1$ to compute the hashed validation dataset $\mathbf{h}_1(D_v)$ and sending it to the aggregator.

Iteration: In each iteration, the following sequence of steps repeats:

1. Aggregator uses the hash function $\mathbf{h}_2$ on two datasets:

Algorithm 1 Description of the protocol.

1: Input: Datasets D_i for $i \in [K]$, validation dataset D_v, Initial seed set D_{init} and a budget p, Parameters: $\{\varrho_v,\{\varrho_{\ell,T}\}_{p\ell=1}\}$.

2: Output: Summary D_s: $D_s \subseteq \cup_{i\in[K]} D_i$ such that $|D_s| = p$.

3: Aggregator is initialized with the validation dataset D_v, and the initial seed summary set D_{init}, i.e. $D_s \leftarrow D_{\text{init}}$.

4: for $\ell = 1 \ldots p$ do

5: if $\ell = 0$ then

6: Aggregator broadcasts $\tilde{\mathbf{g}} = \mathbf{h}_2(\mathbf{h}_1(D_v),\varrho_v)$. 7: end if

8: Aggregator broadcasts $\mathbf{g}_\ell = \mathbf{h}_2(\mathbf{h}_1(D_s),\varrho_{\ell,T})$.

9: for $i = 1 \ldots n$ do

10: Data owner i computes its "bid": $b_i = \max_{x\in D_i} \mathbf{g}_\ell^T\mathbf{h}_1(x) - \tilde{\mathbf{g}}^T\mathbf{h}_1(x)\frac{\ell}{\ell+1}$. Let x_{i^*} be the datapoint corresponding to the bid b_i.

11: Data owner i sends bid b_i to the aggregator. The datapoint is not sent yet.

12: end for

13: Aggregator collects all the bids and chooses the best data owner $i^* = \text{argmax}_i\, b_i$.

14: Aggregator requests the datapoint x_{i^*}, receives it and verifies the bid b_i. Upon verification, $D_s \leftarrow D_s \cup x_{i^*}$.

15: end for
16: return Summary $D_s - D_{\text{init}}$.

---

1) the current summary dataset (initially empty) containing $\mathbf{h}_1$ hashes of data points, 2) hashed validation dataset sent by the consumer. The aggregator then broadcasts these two hashes to all the data providers.

2. Data providers in turn use these hashes to computethe optimal point that provides the maximum marginal gain towards the MMD objective. Data provider *i* computes its marginal improvement value as bid b_i.

3. Aggregator collects only the bids from various participants, and not the actual hashed data points. It then chooses the "winning bid" say b_w and then requests the corresponding hashed data point $\mathbf{h}_1(x_w)$.

4. Finally, the aggregator verifies that the bid value corresponds to the hashed data point provided and if so, adds it to the current summary set.

This process continues until enough points are collected in the summary set. We have shown the following guarantee in our companion work:

Theorem 1 *(Informal) There exists randomized hash functions* $h_2(\cdot)$ *and* $h_1(\cdot)$ *such that the protocol in Algorithm 1 has the following privacy properties:*

a) *For any* $\frac{1}{e} > \delta > \tilde{} 0$*, the releases of the aggregator during Algorithm 1 to any data owner i is* $(\varrho,\tilde{\delta})$*-differentially private over all the iterations/epochs with respect to the datasets* $\cup_{j=6\ i} D_i$*. Similarly, we have* $(\varrho,\tilde{\delta})$*-differentially privacy over all the iterations with respect to validation set* D_v.

b) $J(D_s) \geq (1 - e^{-1})J(\underline{OPT}) - \Delta$*, where OPT is optimal summary and* $\Delta = O(\frac{\log p\sqrt{\ln d}}{\sqrt{d}}) + \frac{1}{\epsilon \log p} < 1$.

c) *The aggregator can at the worst learn only pairwise Eu-clidean distances between points in* D_s.

Remark: The aggregator can only learn pairwise distances between points in $D_s \cup D_v$ because $\mathbf{h}_1$ is random with some privacy properties and its seed is not known to the aggregator. However, this is *not* a formal differential privacy guarantee. It appears that providing formal differential privacy guarantees with respect to the knowledge of the aggregator is extremely difficult or even infeasible as points are being collected for further training in some hashed form.

4. Addressing Trust, Fairness and AuditabilityIssues: Role of Blockchain

Before describing the role of blockchain in addressing the remaining issues, we first note that Hyperledger Fabric [24, 4], provides access control that can be used to restrict the access of hashed data assets to various participants. For example, the data points hashed with $\mathbf{h}_1$ must be accessible to only to the aggregator and the owner of that data point. This will ensure that there is no privacy loss since the hash function $\mathbf{h}_1$ is unknown to the aggregator but known to other participants. Further, aggregator's releases are auditable and it can be ensured that the releases happen through the $\mathbf{h}_2$ function only thereby ensuring privacy of the aggregator releases.

Fair Value Attribution: One natural way of value attribution is to assign value to a data owner that is proportional to the sum of *its* winning bids. The crucial point to note is that, in order to provide auditable/trusted value attribution, we must have the ordered winning bids and their corresponding hashed datapoints. The reason is that the MMD objective is submodular and hence marginal gains (bids) is a function of what has been acquired before. For example, the same data point has a greater value if chosen earlier. Therefore, validating the winning bid and their order is crucial. In fact, we provide for a set of very general value attribution schemes. For example, due to fairness considerations, every participant may agree to choose a re-ordering in which every participant is equally well represented in the first say 30% percent of the points chosen. The final chosen set of points will not change but the value attribution can be made more fair. Such post-process value attribution schemes are possible due to tracking of the protocol's ordering of bids.
In this, we leverage immutable tracking, a key blockchain attribute.

The outstanding concern is ensuring that no data owner cheats. We ensure this through our design which we discuss in the subsequent sections.

5. Architecture Design

The transactional logic of a blockchain application is contained in scripts called *smart contracts* or *chaincode*. Computations performed on blockchain, through these smart contracts, are typically slower, since they must be performed by multiple *endorsing peers*, each potentially querying multiple assets from the ledger and then, finally, reaching a consensus on the computation. This motivates us to design our platform such that most of the computational heavy-lifting is done off the blockchain and keep the blockchain component light-weight. In the next section, we will describe different experiments that have increasing complexity of the blockchain network starting from "no

trust" to "completely trusted" designs. By trust, we mean that an independent auditor can detect any discrepancies in participant actions after the protocol is complete.
We consider three different designs for implementation:

1. No-trust: This is the most basic case where the blockchain component is not present. This case, where there is no trust guaranteed, can be treated as the baseline for performance.

2. Semi-trust: This is a minimal design the comes with some trust guarantees. In this case, it is assumed that the aggregator is trusted and then, it can be ensured that no other participant can cheat.

3. Complete-trust: In this setting, we make no assumptions on the trust-worthiness of any of the participants including the aggregator. This scenario guarantees complete trust, fairness and tamper-free tracking but is the most expensive design from the performance perspective.

Our design involves two main components: an off-chain component and an on-chain component. The off-chain component is responsible for the bulk of the computation that needs to be performed by different participants. The on-chain component is used to record communications between different participants. We will expand on these components below. Refer to the Figure 1.
Off-chain Component: The off-chain component comprises of entities corresponding to a *consumer*, an *aggregator*, *data providers* and optionally, an *auditor*. These entities are implemented as REST servers. Consumer REST server is owned by the consumer and supports services to compute hashed validation set using the hash function $\mathbf{h}_1$ and then record it on the blockchain ledger. Aggregator REST server, owned by the aggregator, supports functions to compute the hash function $\mathbf{h}_2$ on data sets (such as the summary set or the hashed validation dataset), conduct an "auction" operation over the marginal gain bids from different data providers to choose the winner bid, receive & verify the winning hashed datapoint and update the summary to include the new hashed datapoint.

Each data provider owns an instance of the data provider REST server. These instances support APIs for computing the marginal gain bids during each iteration using hashed aggregates from the aggregator and their private training data set. It also supports services to submit hashed point corresponding to the winning bid to the aggregator. All the services offered by above participants record the computed outcomes on the blockchain so that the entire process is immutably tracked and hence is auditable.
On-chain Component: Our blockchain network has four main logical components, that we will describe in detail now.

Organizations: In a blockchain network, organizations represent the various participating entities in the AI marketplace. Thus each of the aggregator, consumer, data provider and auditor, belongs to an organization in the solution. The role of an organization is to enable the network to identify and authenticate respective participants. An organization generates its own cryptographic certificates/keys and owns a set of peers. These peers could include various anchoring peers, endorsing peers or committing peers, and play a central role in endorsement and consensus phases of validating a transaction.

Assets: In a typical permissioned blockchain network such as Hyperledger Fabric, assets refer to data structures that are essential in tracking the underlying process. Collectively, these assets are designed such that they fully capture the provenance graph as shown in Figure 2. Therefore, creation and recording of these assets at various steps of the protocol enables a post-process audit that can exactly recreate the graph in Figure 2. This is immutable and therefore enables verification of the entire process. Our solution includes three generic types of assets that we now describe.

1. HASHEDDATA: As mentioned before, our protocol uses two different random hash functions, i.e., $\mathbf{h}_1$ and $\mathbf{h}_2$. We use the generic HASHEDDATA asset to record the output of these functions applied over different datapoints. Accordingly, there are two concrete realizations of this asset namely HASHEDDATAONE and HASHEDDATATWO to capture outputs from these two functions.

2. BID: This asset captures the details of bids generated by various data providers, in Step 2 of the protocol in Section 3.

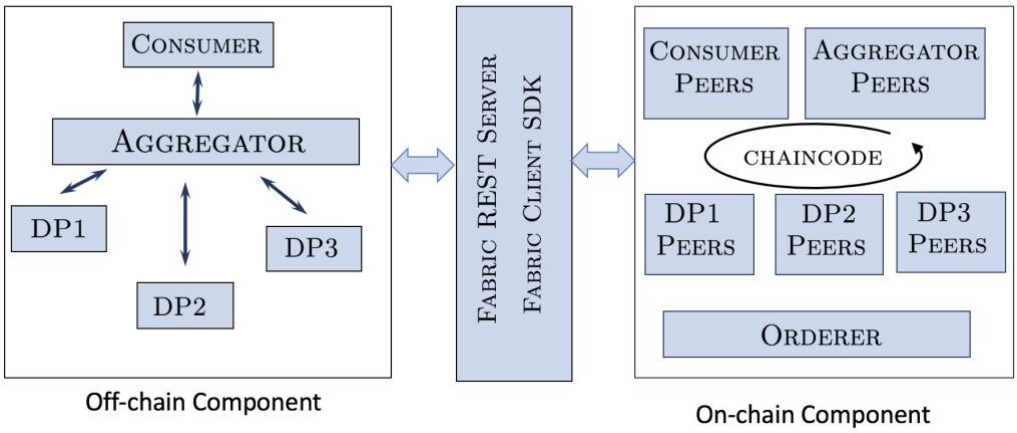

Figure 1. Architecture of Blockchain AI Marketplace

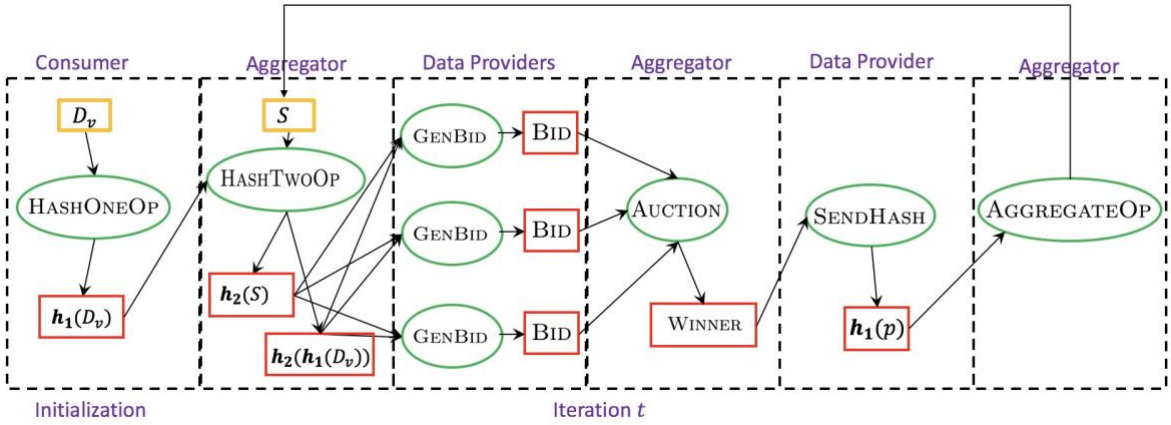

Figure 2. Provenance Graph Captured by the Assets on Blockchain Network

3. GENERICOPERATION: This asset is used to capture the input/output structure of any action performed by various participants, during the protocol. In the provenance graph, these assets capture the links between nodes. This asset is extended to six specific types:

   HASHONEOP: Captures the inputs/outputs and other related details of each $\mathbf{h}_1$ computation. This asset is created in the initialization phase and Step 3 of the protocol. The inputs of this operation include a reference to the input dataset, participant and iteration. The output is an HASHEDDATAONE asset.

   HASHTWOOP: Captures the inputs/outputs and other related details of each $\mathbf{h}_2$ operation. This asset is created in the Step 1 of the protocol in Section 3. The input is an asset HASHEDDATAONE, while the output is a reference to the corresponding HASHEDDATATWO asset generated.

   GENBID: Captures the bid generation operation by different data providers, in Step 2 of the protocol in Section 3. The inputs involved are HASHEDDATATWO assets corresponding to the current summary and hashed validation dataset respectively, while the output captured is a reference to the BID asset.

   AUCTION: Captures the details of auction operation, in Step 3 of protocol in Section 3. The inputs are the BID assets from different data providers, while the output is the BID asset corresponding to the winning bid.

   SENDHASH: Created by data providers, in Step 4 of the protocol in Section 3, to capture the operation of sending out the hashed data point corresponding to the winning bid of the auction operation in each iteration. The input is the winning BID asset and the output is the HASHDATAONE asset corresponding to the winning data provider's data point.

   AGGREGATION: Created by the aggregator, in Step 4 of the protocol in Section 3, to capture the aggregation of the winning hashed data point with the current summary dataset. The inputs include assets HASHEDDATAONE of the current summary and the HASHED-DATAONE of the winning hashed data point and the output is the updated current summary asset HASHEDDATAONE.

Chaincode: In a blockchain network, chaincode contains the asset definitions and logic to operate on them.

Depending on the type of operation invoked by participants, a subset of endorsing peers execute the corresponding functions in the chaincode and generate results. If they agree with each other on the generated results in the assets are modified as per the transaction specification and committed to the blockchain ledger. Although, in general, chaincode could be quite complex, as a guiding principle, we limit our chaincode to simple "update and record" type operations on the above assets. This allows us to delegate the heavy computation to off-chain components, while still ensuring trust, fairness and tamper-free tracking of the protocol.

6. Implementation

We now present the implementation details that would enable our protocol to work in the semi-trust and completetrust scenarios.

Semi-trust: In this scenario, our off-chain components communicate directly with each other via REST API calls, while recording the assets created in the process on the blockchain ledger. As an example, consider the auction operation performed in each iteration to get the incrementally "best" data point. During each such operation, the aggregator broadcasts the $\mathbf{h}_2$ aggregate of the summary dataset to the participants and subsequently creates an asset of type HASHEDDATATWO containing this aggregate value. Each data provider receives this broadcasted aggregate value and uses it to compute its current bid. They record these bids on the blockchain. Subsequently, the aggregator chooses the winning bid, records it on the blockchain and finally, requests the winning data provider for the hashed datapoint corresponding to its bid. The data provider sends out the hashed data point to the aggregator and records the corresponding BID asset to record the hashed datapoint. The key point to note is that such tracking on blockchain will let us figure out if there is any dispute in the communicated assets. This implies that, if the aggregator alone can be trusted, our implementation in this scenario will ensure that no other participant can cheat.

Complete-trust: In this scenario, while the computation is still performed off-chain, the communication happens through queries to blockchain ledger. To illustrate the point, let us re-consider the auction operation and discuss the implementation in this scenario.

As before, the aggregator computes $\mathbf{h}_2$ aggregate of the summary set (off the chain) and records the HASHEDDATATWO asset on the blockchain. The participants in turn query this aggregate from the blockchain ledger and use it to compute the bids. Note that this will allow us to trace exactly what was shared between the data providers and the aggregator in a completely trusted manner.

Subsequently, the computed bids are again recorded on the blockchain, from which they are queried by the aggregator, who in turn chooses the winning bid and records it on the blockchain ledger. The winning data participant is then notified via hyperledger's event handler, who then updates the winning bid with the corresponding hashed data point.

In conclusion, the crucial insight here is that, while in the semi-trust setting the communication between participants is not known, in the complete-trust there is an immutable record of the exact communication between different participants. Thus, for this scenario we completely guarantee that the process is trusted.

Fairness: Given that our process is "trusted" in both the scenarios, under reasonable assumptions in the semi-trust case, we now discuss how one can fairly attribute value to different data providers. For our protocol, we can define "value" of a data point as the marginal improvement in the MMD objective upon its addition. Since the (in fact it's negation) MMD objective is submodular, the value of a datapoint crucially depends on *when* it was chosen. In particular, the value of the point is higher (or at least equal) if it is chosen in the beginning iterations than the later iterations. Thus an immutable order of chosen points is crucial in their value attribution and hence to the notion of *fairness*. One of the key features of blockchain is that chronological order of transactions is preserved in a completely tamper-free manner. Thus, we use blockchain to enable fairness in our solution.

7. Experiments and Results

We now describe the experimental setup and performance evaluation of the three design paradigms explained in Section 6, namely: *no-trust*, *semi-trust* and *completetrust*. The *no-trust* setting has zero "performance overhead" due to the trust constraints, as the blockchain component is empty. On the other hand, in the *complete-trust* setting, participants communicates through the blockchain and therefore, have to wait at multiple steps for the blockchain to commit transactions. Since, committing of transactions involves complex endorsements and consensus protocols, it is expected to be considerably slow. On the positive side, as explained before, this will guarantee trust in an absolute sense. The *semi-trust* setting is a compromise between the two worlds – performance and trust. Here, we use a *fire and forget* approach where participants record their updates on the blockchain and move on without waiting for transactions to commit. The communication happens outside blockchain but participants records their updates on the blockchain. As described before, if the aggregator is assumed to be trusted, this process will guarantee trust with respect to all other participants.

TheExperimentalSetup: Our experimental setup involves a *consumer*, an *aggregator* and three *data providers*. We use a real world dataset from *Allstate insurance company* which was published as part of a Kaggle Competition [1]. We adapt the dataset somewhat to fit our distributed setting. In

an accompanying paper, we report our findings on the accuracy guarantees of our protocol. In this paper, we are interested in the throughput performance of our system under different trust requirements, i.e., *no-trust*, *semi-trust* and *complete-trust*. We briefly describe the dataset here.

The Allstate dataset: The dataset is comprised of insurance data belonging to customers from two states namely, *Florida* and *Connecticut*. We distribute the *Florida* data among all the three data providers uniformly at random. From the *Connecticut* dataset, we take 70% of the data and allocate it completely to the first data provider. This creates a *skew* between the data providers. The validation dataset is comprised exclusively of the *Connecticut* data to simulate the situation that the consumer needs to build a predictive model for customers from that state.

Results: For each of the three scenarios, namely, *no-trust*, *semi-trust* and *complete-trust*, we run several experiments with varying summary set sizes as specified by the set [1000, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 7000]. Our results are shown in the Figure 3.

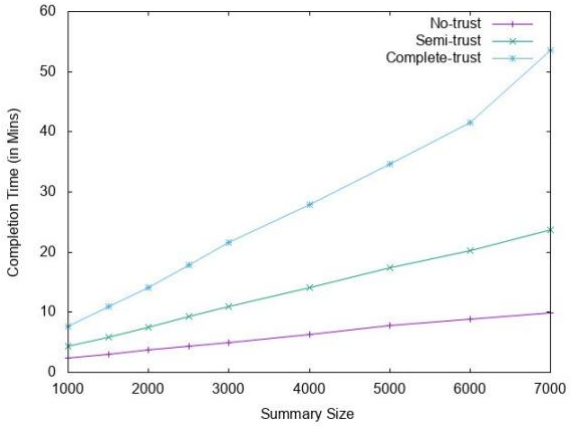

Figure 3. Performance comparison between different trust scenarios.

The results show that if the time taken by the baseline case of *no-trust* scenario is X mins, the *semi-trust* case takes $\approx 2X$ mins and the *complete-trust* scenario takes $\approx 5X$ mins. It should be noted that our protocol is fairly complex from a tracking perspective. Indeed, each iteration of the protocol creates multiple assets and records them on blockchain. In conclusion, we empirically demonstrate that blockchain can be used, at a reasonable overhead, to guarantee different levels of trust, fairness and auditability even in complex machine learning protocols.

8. Related Work

In the machine learning literature, there is a lot of existing work on privacy-preserving classifier training on a single private dataset [13, 7, 23, 22, 14, 5, 12, 29, 2, 20, 27]. One of the most notable in this line of work is the idea of adding noise to stochastic gradient iterations to preserve privacy when learning a model using stochastic gradient descent [23, 2, 22].

When there are many data sources to learn from, an approach called Federated learning [17] was proposed where gradients from a very large number of data sources are compressed and sent to a server where they are fused into a single model update. This focused on communication efficiency. Recent work established that this procedure could be made even differentially private with respect to the various data sources [3]. This line of work is primarily intended on learning a single model from data stored on many edge devices (like mobile phones etc.).

However, our setting differs from the above in various ways: a) We consider a transfer learning setting where a training set needs to be assembled that matches the validation data set. Uniformly randomly drawing data points (like in the above approaches) is not effective b) Ours is in an enterprise context, where the number of sources is not very large, however each source could potentially have very different value attribution that must be evaluated in a trusted manner without compromising on privacy.

Another interesting recent work that is very related on a high level is [25]. In this work, the authors envision blockchain based electronic marketplaces where buyers and sellers can transact with guarantees on trust and privacy. This is geared towards the e-commerce space. Our work is the first step towards enabling an AI marketplace where entities transact model/data assets in a private and trusted manner.

9. Program

```
// SPDX-License-Identifier: MIT
Pragma solidity ^0.8.0;

Import
"@openzeppelin/contracts/token/ERC20/IERC20.sol";
Import "@openzeppelin/contracts/access/Ownable.sol";
Import "@openzeppelin/contracts/utils/Counters.sol";

Contract AIMarketplace is Ownable {
   Using Counters for Counters.Counter;

   Struct Model {
      Uint256 id;
      String name;
      String description;
      Uint256 price;
      Address seller;
      Address buyer;
      Bool isSold;
```

```
        Uint256 escrowAmount;
        Bool isUnderReview;
    }

    IERC20 public paymentToken;
    Counters.Counter private modelCount;
    Mapping(uint256 => Model) public models;

    Event ModelListed(uint256 id, string name, uint256 price, address seller);
    Event ModelPurchased(uint256 id, address buyer, address seller);
    Event EscrowReleased(uint256 id);
    Event EscrowRefunded(uint256 id);
    Event ModelReviewed(uint256 id, uint256 rating, string comment);

    Constructor(IERC20 _paymentToken) {
        paymentToken = _paymentToken;
    }

    Function listModel(string memory _name, string memory _description, uint256 _price) public {
        modelCount.increment();
        uint256 newModelId = modelCount.current();
        models[newModelId] = Model(newModelId, _name, _description, _price, msg.sender, address(0), false, 0, false);
        emit ModelListed(newModelId, _name, _price, msg.sender);
    }

    Function purchaseModel(uint256 _id) public {
        Model storage model = models[_id];
        Require(model.id == _id, “Model does not exist”);
        Require(!model.isSold, “Model is already sold”);

        // Transfer tokens to escrow
        Uint256 escrowAmount = model.price;
        Require(paymentToken.transferFrom(msg.sender, address(this), escrowAmount), “Payment failed”);

        Model.isSold = true;
        Model.buyer = msg.sender;
        Model.escrowAmount = escrowAmount;
        Emit ModelPurchased(_id, msg.sender, model.seller);
    }

    Function releaseEscrow(uint256 _id) public {
        Model storage model = models[_id];
        Require(model.isSold, “Model is not sold”);
        Require(msg.sender == model.seller, “Only seller can release escrow”);

        Require(paymentToken.transfer(model.seller, model.escrowAmount), “Escrow transfer failed”);
        Model.escrowAmount = 0;
        Emit EscrowReleased(_id);
    }

    Function refundBuyer(uint256 _id) public {
        Model storage model = models[_id];
        Require(model.isSold, “Model is not sold”);
        Require(msg.sender == model.buyer, “Only buyer can request refund”);

        Require(paymentToken.transfer(model.buyer, model.escrowAmount), “Refund transfer failed”);
        Model.isSold = false;
        Model.escrowAmount = 0;
        Emit EscrowRefunded(_id);
    }

    Function reviewModel(uint256 _id, uint256 _rating, string memory _comment) public {
        Model storage model = models[_id];
        Require(model.isSold, “Model is not sold”);
        Require(msg.sender == model.buyer, “Only buyer can leave a review”);
        Require(!model.isUnderReview, “Review already submitted”);

        // Logic to save or process the review should go here (e.g., struct for reviews)
        Emit ModelReviewed(_id, _rating, _comment);
        Model.isUnderReview = true;  // Prevent duplicate reviews
    }

    Function getModelDetails(uint256 _id) public view returns (string memory, string memory, uint256, address, address, bool, uint256) {
        Model storage model = models[_id];
        Return (model.name, model.description, model.price, model.seller, model.buyer, model.isSold, model.escrowAmount);
    }
}
```

10. Conclusion

In this work, we have demonstrated the use of blockchain to address issues, such as trust, privacy, fairness in value attribution and auditability, in a collaborative transfer learning setting. This is an enabling use case for the AI Marketplace which is a platform for entities to transact data/models. We demonstrated various design choices for a

blockchain implementation of a transfer learning protocol that trades off trustworthiness with performance in terms of completion time. We showed that design choices that ensures complete trust and partial trust require 5× and 2× more time respectively compared to the baseline that guarantees no trust.

References

[1] https://www.kaggle.com/c/allstate-purchase-predictionchallenge, 2014.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[3] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H Brendan Mcmahan. cpsgd: Communicationefficient and differentially-private distributed sgd. *arXiv preprint arXiv:1805.10559*, 2018.

[4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 464– 473. IEEE, 2014.

[6] Christian Cachin and Marko Vukolic. Blockchain Consensus Protocols in the Wild (Keynote Talk). In *31st International Symposium on Distributed Computing (DISC 2017)*, volume 91, pages 1:1–1:16, 2017.

[7] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069– 1109, 2011.

[8] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gun Sirer, et al. On scaling de-¨ centralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[9] Ariel Ekblaw, Asaph Azaria, John D Halamka, and Andrew Lippman. A case study for blockchain in healthcare:“medrec” prototype for electronic health records and medical research data. In *Proceedings of IEEE open & big data conference*, volume 13, page 13, 2016.

[10] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functionsii. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

[11] Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[12] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. Learning privately from multiparty data. In *International Conference on Machine Learning*, pages 555–563, 2016.

[13] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[14] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1, 2012.

[15] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems*, pages 2280–2288, 2016.

[16] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

[17] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[18] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions-i. *Mathematical Programming*, 14(1):265–294, 1978.

[19] Steve Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI matters*, 1(2):19–21, 2014.

[20] Manas Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*, pages 1876–1884, 2010.

[21] Ali Rahimi and Benjamin Recht. Random features for largescale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[22] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.

[23] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 245–248. IEEE, 2013.

[24] Joao Sousa, Alysson Bessani, and Marko Vukolic. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018.

[25] Hemang Subramanian. Decentralized blockchain-based electronic marketplaces. *Communications of the ACM*, 61(1):78–84, 2017.

[26] Don Tapscott and Alex Tapscott. *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin, 2016.

[27] Abhradeep Guha Thakurta. *Differentially private convex optimization for empirical risk minimization and highdimensional regression*. The Pennsylvania State University, 2013.

[28] Marko Vukolic.´ The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop*

on Open Problems in Network Security, pages 112–125. Springer, 2015.

[29] Xi Wu, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey F Naughton. Differentially private stochastic gradient descent for in-rdbms analytics. *arXiv preprint arXiv:1606.04722*, 2016.

[30] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184.