Brokered Agreements in Multi-Party Machine Learning

Clement Fung
University of British Columbia

Ivan Beschastnikh
University of British Columbia

ABSTRACT

Rapid machine learning (ML) adoption across a range of industries has prompted numerous concerns. These range from privacy (*how is my data being used?*) to fairness (*is this model's result representative?*) and provenance (*who is using my data and how can I restrict this usage?*).

Now that ML is widely used, we believe it is time to rethink security, privacy, and incentives in the ML pipeline by re-considering control. We consider distributed multi-party ML proposals and identify their shortcomings. We then propose *brokered learning*, which distinguishes the *curator* (who determines the training set-up) from that of the *broker* coordinator (who runs the training process). We consider the implications of this setup and present evaluation results from implementing and deploying TorMentor, an example of a brokered learning system that implements the first distributed ML training system with anonymity guarantees.

1 INTRODUCTION

Data has emerged as a premium resource in the modern age of analytics. Entire industries are built on firstly collecting and organizing data, and then computing and deploying machine learning (ML) models for a variety of tasks. *However, in the modern cloud-based architecture, the ML pipeline lives in a single administrative domain.* Although this is efficient, the benefits are one-sided. We propose that the modern ML pipeline fundamentally does not need to be centrally located or even centrally administered. In fact, decomposing the control of the ML pipeline across more than one party leads to a design that benefits all the parties.

As a review, at the most abstract level, the ML pipeline includes the following stages:

APSys '19, August 19–20, 2019, Hangzhou, China

ACM ISBN 978-1-4503-6893-3/19/08…$15.00
https://doi.org/10.1145/3343737.3343744

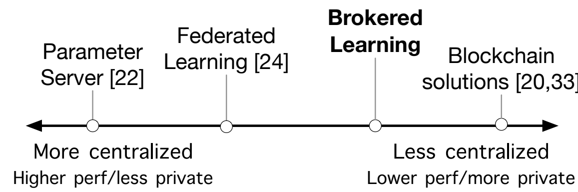

Figure 1: Summary of prior work in distributed ML.

(1) **Collect training data**. A data provider collects training data and houses it in an accessible location. Ideally, the data is collected from a variety of sources to gain as much information as possible to model the expected behavior of the outside world.

(2) **Inner training loop:**

(2a) **Calculate model updates**. In the most general case, a model update is computed on some or all of the collected data. This calculation requires both a set of training data and a view of the current model's state. In this work, we assume the calculation by stochastic gradient descent (SGD) [8].

(2b) **Aggregate and iterate.** The model updates are collected from all calculating sources and aggregated. In online learning or highly parallelized settings, some staleness of model state is acceptable in this process [30], allowing looser consistency models. Once the model updates are applied, the new state of the model is provided for the next iteration, completing the inner loop in the pipeline.

(3) **Deploy model in production.** After a fixed number of iterations, or once convergence heuristics are met, the training loop terminates. The final state of the model is deployed, usually as a service, for use in prediction.

An emerging area in distributed ML is distributed *multi-party* ML, which enables learning from data across a large number of users. In contrast to the centralized data center parameter server model [22], *federated learning* [24] enables multi-party ML by maintaining training data on the provider's device and aggregating model updates at a trusted central coordinator. Federated learning decentralizes the ML pipeline (Figure 1) by having data providers perform the above stages (1) and (2a), while a central coordinator runs stages (2b) and (3). There have been claims of stronger privacy and security in federated learning [7], though recent work has challenged these claims [4, 14, 17, 25].

In the quest to train an optimal model as quickly and efficiently as possible, the central coordinator in federated learning is not incentivized to provide privacy to data providers [29], yet it is most empowered to provide it. The advance of privacy-preserving ML has, so far, been restricted by this view. A single institution administers the entire ML process, and an illusion of control is provided to the data providers.

On the opposite end of centralization, data marketplaces are exchanges that use blockchains to decentralize the ML process (Figure 1). These exchanges facilitate the purchase and exchange of valuable training data (stage 1), and distribute the ML process (stages 2a and 2b) across a blockchain network [20, 33]. These systems use smart contracts [35] to ensure the secure exchange of data and may include methods to appraise training data in a differentially private manner [21]. Since the system lacks trust in any party to perform ML, these systems perform training in trusted execution environments (TEEs) [20] or use cryptographic techniques for ML [33], both of which have high performance overheads [28]. Full decentralization is an extreme proposal, and we believe that intermediate design points on the centralization spectrum (Figure 1) better balance privacy and performance.

A key observation in our work is that *once data providers and model curators agree on a learning objective for multi-party ML, there is no need for the curator to also coordinate the learning*. There is a clear opportunity for a new model for multi-party ML that simultaneously respects the emerging privacy needs of data providers and model utility needs of model curators, while logically centralizing the aggregation stage (2b). We define this new learning setting, called *brokered learning*, in which a neutral third-party coordinates the learning process. We evaluated the plausibility of our brokered learning model by designing one example brokered learning system called *TorMentor* [13]. TorMentor is an anonymous ML system that operates brokered learning over the Tor network [9].

The brokered learning model we propose has well-aligned incentives, which can broaden ML usage even further by pushing parts of the ML pipeline outside of organizations that today control and administer these pipelines. Another advantage of our proposal is to better align modern ML pipelines with new privacy regulations, such as GDPR. For example, brokered learning relieves curators from storing and even observing potentially private user data, both of which are problematic under GDPR [32].

2 TOWARDS BROKERED LEARNING

We recognize that the distributed ML process is made possible by several actors, each providing their own unique value to the system, and each with unique participation incentives.

Data providers contribute the most valuable resource: training data. To train a model that generalizes well to a variety of situations, data should be collected from a variety of users. The contribution of training data for ML is at tension with the rising need for privacy [1, 2]. This has prompted the development of privacy-preserving training methods [15, 34], which allow providers to generate privacy-preserving model updates in stage (2a), prior to the aggregation stage (2b).

An issue with applying these methods in the private multi-party ML setting is that, in a tunable privacy setting, data providers are not incentivized to provide data with lower levels of privacy. As a response, large organizations have implemented their own privacy technologies, which is problematic due to a lack of transparency and potential for implementation errors [36].

On the flip side, when data providers have the freedom to compute model updates locally, there is no process to audit or mandate that the computation is correct. Recent work has shown that this allows malicious data providers to perform attacks on both the shared model and other data providers [4, 14, 17, 25].

Model curators define the desired ML task, and may optionally provide the required algorithms for distributed multi-party ML. In the model above, curators are responsible for stage (3), and optionally may define (but not necessarily perform) stages (2a) and (2b). Curators are incentivized to train the highest performing ML model, and are unconcerned with privacy, which has hindered the deployment of fair and unbiased ML systems [29]. In fact, there is a direct privacy-utility trade-off when it comes to the value of data [10], so providing stronger ML privacy guarantees directly reduces the amount of utility extracted from training data.

The issue of privacy has hindered the ability for untrusting parties to share data and collaborate, forcing organizations to collect massive amounts of training data for their own isolated analysis, and limiting the range of new data domains available to analysts.

Infrastructure providers house the update and aggregation computation (stages 2a and 2b). Functionally, the infrastructure provider does not need to know who is involved, what computation is being executed, or what the model is being used for. They serve as a natural point for brokering and equalizing the incentive interaction between data providers and model curators since they cannot favor either party: in leaking private data, they lose reputation with data providers, and in compromising model utility, they lose model curator business.

Today, a select few curators own massive infrastructures for large scale ML, and the long tail of curators rent infrastructure from these providers. A variety of solutions have proposed the infrastructure provider as a point for introducing privacy guarantees to data providers [7, 28]. But, these models are only available to the largest of infrastructure providers and *assume that the infrastructure provider and the model curator are operated by the same entity.* This creates a setting where private algorithms are proprietary and opaque, and we have seen that implementation errors in privacy-preserving techniques can result in weaker privacy guarantees [36]. If there is no incentive for model curators to provide privacy, why would they do it, and why would they do it well [29]? Thus, we propose *brokered learning*, which builds on the federated learning setting [24], but assumes no trust or prior agreement between data providers and model curators.

3 THREAT MODEL

We assume that the broker is administered by an honest-but-curious neutral party, meaning that it does not initiate actions and follows the prescribed deployment instructions. For example, the broker detects and rejects anomalous behavior and terminates the learning process as instructed by the model curator. A malicious broker could attempt to attack providers or curators, but since this would result in a massive breach of trust and loss of reputation, we do not consider malicious brokers.

We assume that data providers and model curators do not attack the broker itself, rather they aim to attack other curators, other providers, or the outcome of the learning process.

Poisoning attack. In a poisoning attack [6], an adversary meticulously creates adversarial (poisoned) training examples and inserts them into the training data set of a target model. This may be done to degrade the accuracy of the final model (a random attack), or to increase/decrease the probability of a targeted example being predicted as a target class (a targeted attack) [19]. For example, such an attack could be mounted to avoid anomaly detectors [31] or to evade email spam filters [27].

Since clients possess a disjoint set of the total training data in federated learning; they have full control over this set, and can perform poisoning attacks with minimal difficulty, if not audited or verified by an external process.

Information leakage. In an information leakage attack, such as model inversion, an adversary attempts to recover the training examples used to train an ML model by querying crafted examples for the model to predict [11, 12].

Information leakage attacks have been extended to federated learning: instead of querying information from a fully trained model, an adversary observes model updates or infers them from changes in the shared model during the training process [17, 25]. Once they collect a sufficient number of model updates, an adversary can reconstruct training examples that belong to other clients.

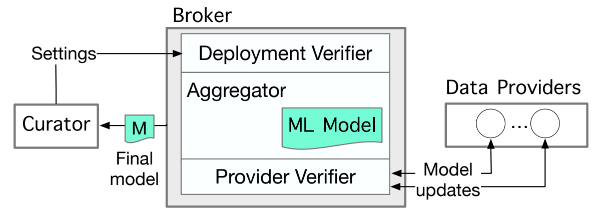

Figure 2: Overview of brokered learning.

Component	Broker function	Possible implementations
Deployment verifier	• Receives deployment requests from model curators • Accepts and enforces curator requirements • Establishes endpoint to train newly specified models	• Smart contract deployment [20] • External auth
Aggregator	• Aggregates model updates from providers • Applies updates to shared model, returns updated state • Evaluates termination cond.	• Total async SGD [30] • Bulk sync SGD [22] • Semi sync SGD [18]
Provider verifier	• Validates providers on join and model update • Rejects malicious provider behavior	• Proof of work [3] • Shapley valuation [21] • Reject on negative influence [5] • External auth

Table 1: A summary of the three broker components.

Because of information leakage attacks, data providers cannot directly expose data or model updates computed on private data. This has motivated a variety of new solutions that protect the privacy of a model update in federated learning, such as secure aggregation [7] and differentially-private federated learning [15].

Sybil attacks. Since data providers join the system anonymously, they can generate sybils, or multiple colluding virtual clients, to attack the system [14]. Because of this, a method for verifying or auditing the identity of data providers is critical for multi-party ML. This can either be performed with external authorization or through other sybil-resilient mechanisms common in modern blockchains [3, 16, 26].

4 BROKERED LEARNING DETAILED

Brokered learning relieves the incentive tension in federated learning by allowing model curators to parameterize the components that interface data providers with the shared model. These components are shown in Figure 2 and their roles are defined in Table 1, with a list of suggestions for each component observed in prior work.

Curators define the deployment parameters in brokered learning: the model type, the learning task, and the services for provider and model update verification. By defining these parameters, the model curator dually ensures model utility and thwarts poisoning attacks from data providers. This brokered deployment can be performed securely using parameters to a smart contract [20, 35] or via a trusted service.

Data providers contribute data to the ML task and control their participation criteria. Instead of fully trusting the curator, as they would in federated learning, providers coordinate the learning process with a trusted broker.

Brokered learning allows these providers to contribute to a shared global model, without being aware of nor trusting each other. Providers interface with the broker by iteratively requesting access to the system and sending a corresponding model update. Brokered learning supports a variety of synchrony models, including total asynchronous [30], bulk synchronous [22], and hybrid SGD models [18].

While training through SGD, each provider can use their personal privacy parameters [15] and is not obligated to reveal more information than needed for successful ML.

A broker is a short-lived process that coordinates the training. The broker exposes interfaces that are responsible for brokering the agreement between data provider and model curator and resolves their tension by dividing their APIs: as long as model curators specify validation services that adequately ensure utility, and data providers send model updates that adequately provide privacy, both parties will be satisfied. As shown in Figure 2, these interfaces sit as protective and expressive layers on top of the machine learning model that would otherwise be vulnerable and non-negotiable in federated learning.

In our service-based vision, brokers are not intended to be long lasting, and their sole function should be to broker the specific agreement between users to facilitate multi-party ML of a single model. Brokers may be explicitly managed by governments, blockchains or businesses, all of whom are incentivized to provide privacy, anonymity and fairness in distributed ML.

5 TORMENTOR DESIGN

Here we overview TorMentor's design; a more complete description is available in prior work [13]. Since data providers and model curators no longer have to directly interact to perform multi-party ML, we envision an *extreme* example that showcases a novel setting that would be impossible under the current model of federated learning: *anonymous* multi-party ML. In this setting, data providers and model curators interact through an anonymous marketplace for distributed multi-party ML.

We built TorMentor, an example brokered learning system that realizes this novel anonymous setting. In lieu of trusted cloud infrastructure or a governing organization, brokers are run as hidden Tor services [9]. Data providers and model curators communicate with a hidden Tor service endpoint, which satisfy the roles defined in Table 1 and expose a minimal API [13].

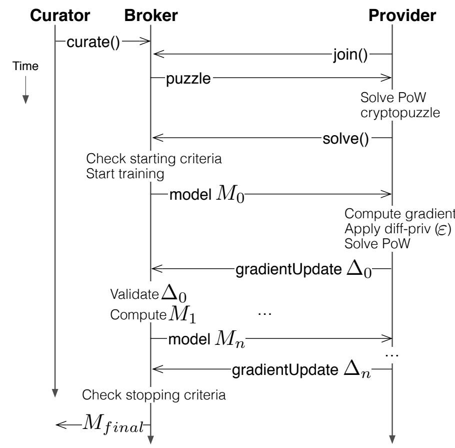

Figure 3: Overview of the TorMentor protocol.

Design overview. Each TorMentor broker is deployed as a Tor hidden service with a unique and known *.onion* domain. As in brokered learning, data providers may join the learning process if they satisfy the validation requirements defined by the model curator. Each broker is associated with a pool of providers that perform SGD.

Without existing reputation scores or trust between brokers and data providers, the curator defines a *validator* service to ensure the integrity of model updates sent by providers to the broker. The validator uses 2 elements to verify the integrity of provider activity in the system. First, a validation dataset is used to verify a proportion of the incoming stream of model updates. This dataset is provided by the model curator and used as the ground truth for the model; any update that causes a significant degradation in validation accuracy will be rejected in a Reject on Negative Influence (RONI) [5]. Secondly, the validator exposes a cryptographic proof of work puzzle [3] that providers are required to solve before joining the system and again when submitting a model update. The difficulty of this puzzle increases when providers fail a RONI test; this alleviates the risk of sybils by significantly increasing the cost of sybil-based poisoning attacks [14]. As a default, we validate 10% of all model updates and use an initial difficulty of 3. When a provider fails a RONI validation, their puzzle difficulty increases by 1.

Curator API. The curator uses the **curate** call (Figure 3) to bootstrap a new model by defining a *common learning objective*: the model type, the desired training interface and a validation service, as described above.

For each model defined by a curator, a single broker is created and deployed as a hidden service and the system waits for providers to contact the service with a message to join. In TorMentor, a RONI validation dataset is used as an example of a mechanism for rejecting adversaries, but can be replaced with any curator desired requirements, in a programmable form such as a smart contract [35].

Provider API. A provider uses the **join** call (Figure 3) to join a curated model. A provider's data is validated against the objective when joining. Our prototype only checks that the size of the model update matches those of the curator dataset, but a differentially-private method for data valuation [21] can also be used to verify provider integrity.

To comply with proof-of-work validation, the provider uses the **solve** call to join the model and submit model updates, similar to that of the Bitcoin [26] protocol, in which a cryptographic SHA-256 admission hash is inverted and the solution is verified, creating a new puzzle once published [3]. Once the proof-of-work is completed, the provider is accepted as a contributor to the model. Once the desired number of providers have been accepted, collaborative model training is performed through the brokered learning protocol: each provider computes their SGD update on the global model and pushes it to the parameter server through the **gradientUpdate** function, which models the process of federated learning.

Protecting providers and curators. Since providers compute gradient updates *locally*, providers maintain a personal privacy level ε when calculating differentially-private updates during model training [15]. Some providers may value privacy more than others and thus will tune their own privacy risk, while curators want to maximize their model utility. In brokered learning, this is handled through the curator-defined validation service. Model updates that provide insufficient model utility will be rejected, keeping both the data provider and the model curator safe. *TorMentor is the first system to support anonymous ML in a setting with heterogeneous user-controlled privacy goals.*

6 EVALUATION

Credit card dataset. In our evaluation we envision multiple credit card companies collaborating to train a model that predicts defaults on credit card payments. However, the information in the dataset is private to each credit card company. In this context, a credit agency can act as the curator, the broker is a commercial trusted service provider, and data providers are the credit card companies.

To evaluate this use-case we used a credit card dataset [37] from the UCI machine learning repository [23]. The dataset has 30,000 examples and 24 features. The features represent information about customers, including their age, gender and education level, along with information about the customer's payments over the last 6 months. The dataset also contains information about whether or not the given customer managed to pay their next credit card bill, which is used as the labeled output for the model.

Prior to training, we normalized, permuted, and partitioned the datasets into a 70% training and 30% testing shard. For each experiment, the training set is further sub-sampled to create a single data provider's dataset, and the testing shard is used as the curator-provided validation set. Training error, our primary metric, is calculated as the error when classifying the entire 70% training shard. However, note that in brokered learning no single data provider would have access to the entire training dataset.

Wide-area (WAN) deployment on Azure. We evaluated brokered learning at scale by deploying TorMentor on a geo-distributed set of 25 Azure VMs, each running in a separate data center, spanning 6 continents. Tor's default `stretch` distribution was installed on each VM. We deployed the broker at our home institution as a hidden Tor service. The median ping latency (without Tor) from the VMs to the broker was 133.9ms with a standard deviation (SD) of 61.9ms. With Tor, the median ping latency was 715.9ms with a SD of 181.8ms. We evenly distribute a varying number of data providers across the 25 VMs. Each provider joins the system with a bootstrapped sample of the original training set (n = 21,000 and sampled with replacement), and participates in asynchronous model training.

6.1 Scalability and overhead

We evaluated TorMentor's scalability by varying the number of data providers. We evaluate the latency overhead by deploying a new broker and initializing the training process once all providers have joined. All nodes were honest, held a subsample of the original dataset, and performed asynchronous SGD.

Figure 4 shows that, when updating asynchronously, the model convergences at a faster rate as we increase the number of providers.

We also compared the convergence time on TorMentor with a baseline brokered learning instance (which is identical but bypassed Tor). This models a data marketplace in which anonymity is not a concern, but users still do not want to share their data. On average, the overhead incurred from using Tor ranges from 5-10x. For example, with 200 data providers TorMentor finishes training in 67s and without Tor it takes 13s.

6.2 Poisoning defenses evaluation

We evaluate the ability of RONI and proof of work as validation processes in brokered learning when defending against random poisoning attacks. To do this, we deployed TorMentor in a setting with 8 providers. We then included malicious

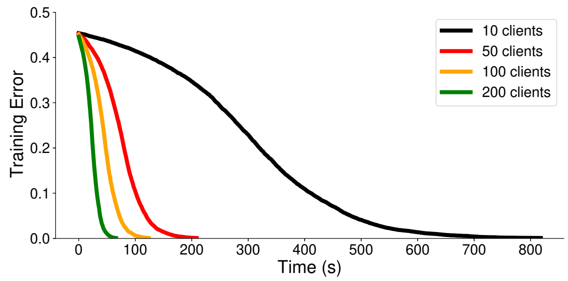

Figure 4: TorMentor model convergence in deployments with 10, 50, 100, and 200 data providers.

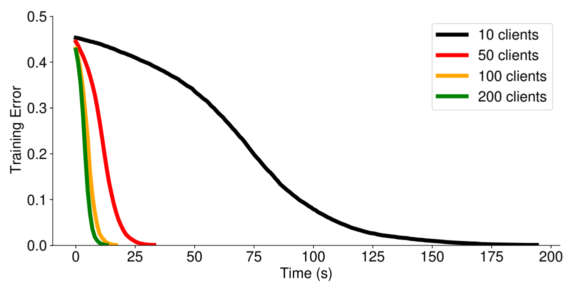

Figure 5: Model convergence in TorMentor without Tor with varying number of data providers.

providers with label flipped data [6] and varied both the proportion of malicious providers in the system and the required drop in model influence for a flagged RONI validation. Each time a provider failed a RONI validation, the difficulty of their proof of work puzzle was increased by 1. Figure 6 shows the training error for the first 250s for a RONI threshold of 2%, while varying the proportion of poisoning attackers from 25% to 75% and validating 10% of model updates.

As the number of poisoners increases, different effects can be observed. When the number of poisoners is low (below 25%), the model still converges, but slower than normal. With 50% poisoning, the model begins to move away from the optimum, but is successfully defended by the provider validator, which increases the proof of work required for all of the poisoners within 30s. From this point, the poisoners struggle to outpace the honest nodes, and the model continues on a path to convergence. Lastly, when the proportion of poisoners is 75%, the increase in proof of work is too slow to react; the model accuracy is compromised within 20s and struggles to recover.

Figure 7 shows the execution of model training with 50% poisoning providers for different RONI validation thresholds. As the threshold decreases, adversaries are removed from the system more quickly, allowing the model to recover from the poisoning damage. Setting the RONI threshold too low

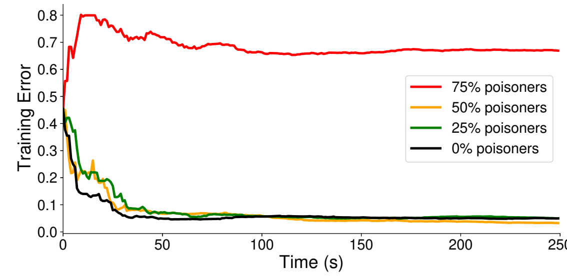

Figure 6: Training error over time while attacked by a varying fraction of poisoners. RONI threshold is 2%.

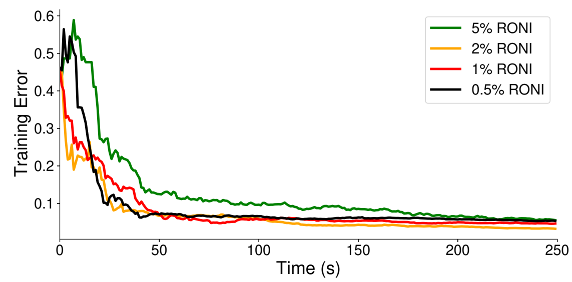

Figure 7: Training error over time, when attacked by 50% poisoners. RONI threshold is varied 0.5% – 5%.

is dangerous as it increases the effect of false positives: Figure 7 shows that the model initially performs poorly due to incorrectly penalizing honest providers.

From this evaluation, we note that, if a poisoner was able to detect this defense, and attempt to leave and rejoin the model, an optimal proof of work admission puzzle should require enough time such that this strategy becomes infeasible.

This evaluation shows that even a simple heuristic, such as validation error, can be effective in verifying the integrity of model updates sent by data providers in brokered learning. In practice, a model curator can supply an arbitrary function to validate providers, using elements like external reputation scores or data valuation [21].

7 CONCLUSION

We are increasingly relying on ML for our everyday activities, yet the ML training process is highly centralized. In this paper we proposed brokered learning as the next step in evolving federated learning: decoupling the role of the curator that defines the model, from the aggregator that trains the model. We also described the design of TorMentor, an example brokered learning system, that pushes the limits of multi-party ML by providing anonymity to curators and data providers through Tor. We hope that learning inspires further research in privacy-preserving ML systems that better consider the incentives of data providers and model curators.

REFERENCES

[1] April 24, 2019. Facebook sets aside $3bn for privacy probe. https://www.bbc.com/news/business-48045138

[2] January 21, 2019. Google hit with £44m GDPR fine over ads. https://www.bbc.com/news/technology-46944696

[3] Adam Back. 2002. *Hashcash - A Denial of Service Counter-Measure.* Technical Report.

[4] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. 2018. How To Backdoor Federated Learning. *ArXiv e-prints* (2018). arXiv:cs.CR/1807.00459

[5] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. 2010. The Security of Machine Learning. *Machine Learning* 81, 2 (2010).

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks Against Support Vector Machines. In *Proceedings of the 29th International Conference on International Conference on Machine Learning.*

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[8] Léon Bottou. 2010. *Large-Scale Machine Learning with Stochastic Gradient Descent.*

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-generation Onion Router. In *Proceedings of the 13th Conference on USENIX Security Symposium.*

[10] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014).

[11] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[12] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *USENIX SEC.*

[13] Clement Fung, Jamie Koerner, Stewart Grant, and Ivan Beschastnikh. 2018. Dancing in the Dark: Private Multi-Party Machine Learning in an Untrusted Setting. *arXiv e-prints* (2018). arXiv:cs.CR/1811.09712

[14] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2018. Mitigating Sybils in Federated Learning Poisoning. *ArXiv e-prints* (2018). arXiv:cs.LG/1808.04866

[15] Robin C. Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. *NIPS Workshop: Machine Learning on the Phone and other Consumer Devices* (2017).

[16] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP).*

[17] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.*

[18] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *NSDI.*

[19] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. 2011. Adversarial Machine Learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (AISec).*

[20] Nick Hynes, David Dao, David Yan, Raymond Cheng, and Dawn Song. 2018. A Demonstration of Sterling: A Privacy-preserving Data Marketplace. *Proceedings of the VLDB Endowment* 11, 12 (2018).

[21] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. 2019. Towards Efficient Data Valuation Based on the Shapley Value. In *Proceedings of the 22th International Conference on Artificial Intelligence and Statistics (AISTATS).*

[22] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14).*

[23] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[24] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics.*

[25] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. 2018. Inference Attacks Against Collaborative Learning. *ArXiv e-prints* (2018). arXiv:cs.CR/1805.04049

[26] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. (2009).

[27] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. 2008. Exploiting Machine Learning to Subvert Your Spam Filter. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats.*

[28] Olga Ohrimenko, Felix Schuster, Cedric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. 2016. Oblivious Multi-Party Machine Learning on Trusted Processors. In *USENIX SEC.*

[29] Rebekah Overdorf, Bogdan Kulynych, Ero Balsa, Carmela Troncoso, and Seda Gürses. 2018. Questioning the assumptions behind fairness solutions. *NeurIPS Workshop: Critiquing and Correcting Trends in Machine Learning* (2018).

[30] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 24.*

[31] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. 2009. ANTIDOTE: Understanding and Defending Against Poisoning of Anomaly Detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement.*

[32] Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. 2019. The Seven Sins of Personal-Data Processing Systems under GDPR. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19).*

[33] Muhammad Shayan, Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2018. Biscotti: A Ledger for Private and Secure Peer-to-Peer Machine Learning. *ArXiv e-prints* (2018). arXiv:1811.09904

[34] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[35] Nick Szabo. 1997. Formalizing and Securing Relationships on Public Networks. *First Monday* 2, 9 (1997).

[36] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. 2017. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12. (2017).

[37] I-Cheng Yeh and Che-hui Lien. 2009. The Comparisons of Data Mining Techniques for the Predictive Accuracy of Probability of Default of Credit Card Clients. *Expert Systems with Applications: An International Journal* 36, 2 (2009).