Universal Multi-Party Poisoning Attacks

Saeed Mahloujifar* Mohammad Mahmoody† Ameer Mohammed‡

November 4, 2021

Abstract

In this work, we demonstrate universal multi-party poisoning attacks that adapt and apply to any multi-party learning process with arbitrary interaction pattern between the parties. More generally, we introduce and study (k, p)-poisoning attacks in which an adversary controls $k \in [m]$ of the parties, and for each corrupted party P_i, the adversary submits some poisoned data $\mathcal{T}_i'$ on behalf of P_i that is still "$(1-p)$-close" to the correct data $\mathcal{T}_i$ (e.g., $1-p$ fraction of $\mathcal{T}_i'$ is still honestly generated). We prove that for any "bad" property B of the final trained hypothesis h (e.g., h failing on a particular test example or having "large" risk) that has an arbitrarily small constant probability of happening without the attack, there always is a (k, p)-poisoning attack that increases the probability of B from μ to by $\mu^{1-p\cdot k/m} = \mu + \Omega(p \cdot k/m)$. Our attack only uses clean labels, and it is online.

More generally, we prove that for any bounded function $f(x_1, \ldots, x_n) \in [0, 1]$ defined over an n-step random process $\mathbf{x} = (x_1, \ldots, x_n)$, an adversary who can override each of the n blocks with *even dependent* probability p can increase the expected output by at least $\Omega(p \cdot \mathrm{Var}[f(\overline{\mathbf{x}})])$.

Contents

1 Introduction **2**
1.1 Our Contribution 2
1.2 Technical Overview 4

2 Multi-Party Poisoning: Definitions and Main Results **6**

3 Multi-Party Poisoning via Generalized p-Tampering **8**
3.1 Proving Theorem 3.5 10
3.1.1 Warm up: Computationally Unbounded Adversaries 11
3.1.2 Proving Theorem 3.5 for Polynomially Bounded Attacks 13
3.2 Obtaining (k, p)-Poisoning: Proof of Theorem 2.5 using Theorem 3.5 16

A Some Useful Inequalities **20**
A.1 Relating the Bias to the Variance 20

*This is a full version of a paper that was previously published in the Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

*sfar@princeton.edu

†mohammad@virginia.edu

‡ameer.mohammed@ku.edu.kw

1 Introduction

Learning from a set $\mathcal{T} = \{d_1 = (a_1, b_1), \ldots, d_n = (a_n, b_n)\}$ of training examples in a way that the predictions generalize to instances beyond $\mathcal{T}$ is a fundamental problem in learning theory. The goal here is to produce a hypothesis h in such a way that $h(a)$, with high probability, predicts the "correct" label b, where the pair $(a, b) = d$ is sampled from the target (test) distribution $\mathbf{d}$. In the most natural setting, the examples in the training data set $\mathcal{T}$ are also generated from the same distribution $\mathbf{d}$, however this is not always the case (e.g., due to noise in the data).

Poisoning attacks. Many previous works studying noise in the data allow it to be *adversarial* and *maliciously* chosen against the learner [Val85, KL93, BEK02]. A tightly related and more recent approach to the problem of learning under adversarial noise is the framework of so-called *poisoning* (aka *causative*) attacks [BNS+06, BNL12, PMSW16], in which the adversary's goal is not necessarily to completely prevent the learning, but perhaps it simply wants to increase the risk of the hypothesis produced by the learning process or make it more likely to fail on a particular test instance (i.e., getting a *targeted* poisoning attack [BNS+06, STS16]).

Multi-party poisoning. In the distributed setting [MR17, MMR+16, BIK+17, KMY+16], the training data $\mathcal{T}$ might be coming from various sources; e.g., it can be generated by m data providers $P_1, \ldots, P_m$ in an online way, while at the end a fixed algorithm, called the aggregator G, generates the hypothesis h based on $\mathcal{T}$. The goal of $P_1, \ldots, P_m$ is to eventually help G construct a hypothesis h that does well (e.g. in the case of classification) in predicting the label b of a given instance a, where $(a, b) \leftarrow \mathbf{d}$ is sampled from the final test distribution. The data provided by each party P_i might even be of "different type", so we cannot simply assume that the data provided by P_i is necessarily sampled from the same distribution $\mathbf{d}$. To model this more general setting, we let $\mathbf{d}_i$ model the distribution from which the training data $\mathcal{T}_i$ (of P_i) is sampled. Poisoning attacks can naturally be defined in the distributed setting as well [FYB18, BVH+18, BGS+17, HO18] to model adversaries who partially control the training data $\mathcal{T}$. These works, however, focus on attacking and defending specific learning tasks. This leads us to the central question of this work.

What is the inherent *provable power of poisoning attacks in the multi-party setting?*

Answering the above question is critical for understanding the *limits* of provable security against multi-party poisoning.

1.1 Our Contribution

We first formalize a new general model multi-party poisoning. We then prove the existence of universal data poisoning attacks in the multi-party setting that apply to any task.

New attack model: (k, p)-poisoning attacks. our first contribution of this work is to formalize a general notion that covers multi-party poisoning attackers that corrupt k out of m data provider parties and furthermore, for each message sent by a corrupted party, the adversary still generates data that is "close" to the honestly generated data. More formally, a (k, p)-poisoning attacker Adv can first choose to corrupt k of the parties. Then, if a corrupted $\widetilde{P}_i$ is supposed to send the next message, then the adversary will sample $d \leftarrow \widetilde{\mathbf{d}}$ for a maliciously chosen distribution $\widetilde{\mathbf{d}}$ that is

guaranteed to be p to the original distribution $\mathbf{d}_i$ in total variation distance. Our (k, p)-poisoning attacks include the so called "p-tampering" attacks of [MDM18b] as special case by letting $k = m$ (m is the number of parties). Moreover, (k, p)-attacks also include the standard model of k static corruption in secure multi-party computation (in cryptography) letting $p = 1$. Our main result in this works is to prove the *universal* power of (k, p)-poisoning as follow. We show that in *any* m-party learning protocol, there exist a (k, p)-poisoning adversary that increases probability of the produced hypothesis h having a bad property B (e.g., failing on a particular target instance known to the adversary). For the formal version of Theorem 1.1, see Theorem 2.5.

Theorem 1.1 (Power of (k, p)-poisoning attacks–**informal**). *Let $\Pi = (P_1, \ldots, P_m)$ be an m-party learning protocol for an m-party learning problem. Also let B be a bad property defined over the output of the protocol. There is a polynomial time (k, p)-poisoning attack such that, given oracle access to the data distribution of the parties, it can increase the probability of B from μ to $\mu^{1-kp/m}$.*

Example. By corrupting half of the parties (i.e., $p = 1, k = m/2$) the adversary can increase the probability of any bad event B from 1/100 to 1/10.

Universal nature of our attack. Our attacks are *universal* in the sense that they could be applied to *any* learning algorithm for *any* learning task, and they are *dimension-independent* as they applied to any data distribution. On the other hand, our universal attacks rely on an initial vulnerability of arbitrary small *constant* probability that is then amplified through the poisoning attack. As a result, although recent poisoning attacks (e.g., see [KSL18]) obtain *stronger* bounds in their attack against specific defenses, our attacks apply to *any* algorithm with any built in defenses.

Deriving attacks on federated learning as special case. Since we allow the distribution of each party in the multi-party case to be completely dependent on that party, our attacks cover the case of model poisoning in federated learning [BVH$^+$18, BCMC18], in which each party sends something *other* than their plain share of data, as special case. In fact, multiple works have already demonstrated the power of poisoning attacks and defences in the federated learning setting (e.g., see [FYB18, BCMC18, CWCP18, CSX17, GR$^+$18, YCRB18, TCC19, CV19, HZ19]). Some of these attacks obtain stronger quantitative bounds in their attacks, however this is anticipated as these works investigate attacks on *specific* learners, while a crucial property of our attack is that our attacks come with *provable* bounds and are *universal* in that they apply to *any* learning task and *any* hypothesis class (including neural nets as special case), if there is an initial $\Omega(1)$ vulnerability (for some bad property) over the generated hypothesis.

Note that, our attacks actually do not need the exact history of examples that are used by parties, and only need to know the updates sent by the parties during the course of protocol. Suppose an uncorrected party randomizes its local model (e.g., for differential privacy purposes) and shares an update u_i with the server. Knowledge of u_i is enough for our attacker. One might go even further and ask what if the updates are sent in a secure/private way? Interestingly, our attack work in that model too as it only needs to know the effect of the updates on the central model at the end of round $i - 1$ (because all the attack wants is a random continuation of the intermediate model).

It also worth mentioning that our attack requires sampling oracles from distributions of all the parties. This might seem that we are giving the adversary too much power. However, we think the right way to define security of federated learning is by giving the adversary everything that hat might be leaked to them. This way of defining security is inspired by cryptography. For instance,

when modeling the chosen plaintext security of encryption schemes, adversary is given access to an encryption oracle, while one might question how realistic it is. Analogously, In federated learning, the adversary can potentially gather some statistics about the distribution of other parties and learn them over time. However, as mentioned above, we do not need to give adversary access to the actual data of honest parties. Only the public effect of them on the shared model is needed.

Further Related Work. Recent breakthroughs of Diakonikolas et al. [DKK+16] and Lai et al. [LRV16] demonstrated the surprising power of algorithmic robust learning over poisoned training data with limited risk that does not depend on the dimension of the distribution (but still depends on the fraction of poisoned data). These works led to an active line of work (e.g., see [CSV17, DKS17, DKS18a, DKK+18, PSBR18, DKS18b, SKL17] and references therein) exploring the possibility of robust statistics over poisoned data with algorithmic guarantees. The works of [CSV17, DKS18a], followed by [BBV08], performed *list-decodable* learning, and [BDLS17, DKK+18, PSBR18] studied supervised learning.

On the negative side, Mahloujifar, Mahmoody and Diochnos [MM17, MDM18a] studied (universal) poisoning attacks that apply to any learning task and any hypothesis class and showed that such attacks can indeed increase the error of any classifiers for any learning problem by a constant probability, so long as there is an initial constant error probability. The attack model used in [MM17, MDM18a], called p-tampering, was a generalization of a similar model introduced in Austrin et al. [ACM+14] in the bitwise setting in a cryptographic context. These attacks (like the ones in our work) were *universal* in the sense that they could be applied to *any* learning algorithm for *any* learning task, and *dimension-independent* as they applied to any data distribution. On the other hand, these universal attacks rely on an initial vulnerability of arbitrary small *constant* probability that is then amplified through the poisoning attack. That is why such universal attacks (including those in the multi-party setting) are not in contradiction with the above results.

1.2 Technical Overview

Previous universal poisoning attacks of [MM17, MDM18a] for the *single* party case are designed in a setting in which each training example is chosen by the adversary with *independent* probability p. We first describe where exactly the ideas of these works come short of extending to the multiparty case, and then we explain how to borrow ideas from attacks on coin-tossing protocols in cryptography [BOL89, HO14] and obtain the desired attacks of this work.

p-tampering attacks and their shortcoming. For starters, let us assume that the adversary gets to corrupt and control k *randomly* selected parties. In this case, it is easy to see that, at the end every single message in the protocol Π between the parties $P_1, \ldots, P_m$ is controlled with exactly probability $p = k/m$ by the adversary Adv (even though these probabilities are correlated). Thus, at a high level it seems that we should be able to use the p-tampering attacks of [MM17, MDM18a] to degrade the quality of the produced hypothesis. However, the catch is that the proof of p-tampering attacks of [MM17, MDM18a] (and the bitwise version of [ACM+17]) crucially rely on the assumption that each message (which in our context corresponds to a training example) is tamperable with *independent* probability p, while corrupting k random parties, leads to tamperable messages in a correlated way.

We prove our main results by first proving a general result about the power of "biasing" adversaries whose goal is to increase the expected value of a random process by controlling each

incoming "segment" (aka block) of the random process with probability q (think of q as $\approx p \cdot k/m$). These segments/blocks correspond to single or multiple training examples shared during the learning. As these biasing attacks generalize p-tampering attacks, we simply call them *generalized* p-tampering attacks. We now describe this attack model and clarify how it can be used to obtain Theorem 1.1.

Generalized p-tampering: new model for biasing attacks. In this work we introduce generalized p-tampering (biasing) attacks that are defined for any random process $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and a function $f(\overline{\mathbf{x}}) \in [0,1]$ defined over this process. In order to explain the attack model, first consider the setting where there is no attacker. Now, given a prefix $x_1, \ldots, x_{i-1}$ of the blocks, the next block x_i is simply sampled from its conditional probability distribution $(\mathbf{x}_i \mid x_1, \ldots, x_{i-1})$. (Looking ahead, think of x_i as the i'th training example provided by one of the parties in the interactive learning protocol.) Now, imagine an adversary who enters the game and whose goal is to increase the expected value of a function $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ defined over the random process $\overline{\mathbf{x}}$ by tampering with the block-by-block sampling process of $\overline{\mathbf{x}}$ described above. Before the attack starts, there will be a a list $S \subseteq [n]$ of "tamperable" blocks that is *not* necessarily known to the Adv in advance, but will become clear to him as the game goes on. Indeed, this set S itself will be first sampled according to some fixed distribution $\mathbf{S}$, and the crucial condition we require is that $\Pr[i \in \mathbf{S}] = p$ holds for all $i \in [n]$. After $S \leftarrow \mathbf{S}$ is sampled, the sequence of blocks $(x_1, \ldots, x_n)$ will be sampled block-by-block as follows. Assuming (inductively) that $x_1, \ldots, x_{i-1}$ are already sampled so far, if $i \in S$, then Adv gets to fully control x_i and determine its value, but if $i \notin S$, then x_i is simply sampled from its original conditional distribution $(\mathbf{x}_i \mid x_1, \ldots, x_{i-1})$. At the end, the function f is computed over the (adversarially) sampled sequence.

We now explain the intuitive connection between generalized p-tampering attacks and (k,p)-poisoning attacks. The main idea is that we will use a generalized q-tampering attack for $q = p \cdot k/m$ over the random process that lists the sequence of training data provided by the parties during the protocol. Let $\mathbf{S}$ be the distribution over $[n]$ that picks its members through the following algorithm. First choose a set of random parties $\{Q_1, \ldots, Q_k\} \subseteq \{P_1, \ldots, P_m\}$, and then for each message x_j that belongs to Q_i, include the corresponding index j in the final sampled $S \leftarrow \mathbf{S}$ with independent probability p. It is easy to see that $\mathbf{S}$ eventually picks every message with (marginal) probability $q = p \cdot k/m$, but it is also the case that these inclusions are not independent events. Finally, to use the power of generalized p-tampering attacks over the described $\mathbf{S}$ and the random process of messages coming from the parties to get the results of Theorem 1.1, roughly speaking, we let a function f model the loss function applied over the produced hypothesis. Therefore, to prove Theorem 1.1 it is sufficient to prove Theorem 1.2 below which focuses on the power of generalized p-tampering biasing attacks.

Theorem 1.2 (Power of generalized p-tampering-**informal**). *Suppose $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a joint distribution such that, given any prefix, the remaining blocks could be efficiently sampled in polynomial time. Also let $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. Then, for any set distribution $\mathbf{S}$ for which $\Pr[i \in \mathbf{S}] = p$ for all i, there is a polynomial-time generalized p-tampering attack (over tampered blocks in $\mathbf{S}$) that increases the average of f over its input from μ to (arbitrarily close to) $\mu' = \mu^{-p} \cdot \mathbb{E}[f(\overline{x})^{1+p}]$. In particular, if f is boolean function, then it hold that $\mu' = \mu^{1-p}$.*

The formal statement of Theorem 1.2 above follows from Theorem 3.5.

Bitwise vs. blockwise attacks. It is easy to see that in the definition of generalized p-tampering attacks, it does not matter whether we define the attack bit-by-bit or block-by-block. The reason is

that, even if we break down each block into smaller bits, then still each bit shall eventually fall into the set of tamperable bits, and the model allows correlation between the inclusion and exclusion of each block/bit into the final tamperable set. This is in contrast to the p-tampering model for which this equivalence is not true. In fact, optimal bounds achievable by bitwise p-tampering as proved in [ACM+17] are *impossible* to achieve in the blockwise p-tampering setting [MM17]. Despite this simplification, we still prefer to use a blockwise presentation of the random process, as this way of modeling the problem allows better tracking measures for the attacker's sample complexity.

Ideas behind the tampering attack of Theorem 1.2. To prove Theorem 1.2 we use ideas from [HO14, BOL89] in the context of coin-tossing attacks and generalize them using new techniques to obtain our generalized p-tampering attacks.

Rejection sampling attack. The simplified version of our attack can be described as follows. Based on the nature of this attack, we call it the "rejection sampling" (RS) attack. For any prefix of already sampled blocks $(x_1, \ldots, x_{i-1})$, suppose the adversary is given the chance of controlling the next i'th block. The RS tampering then works as follows:

1. Let $x'_i, \ldots, x'_n$ be a random continuation of the random process, conditioned on $(x_1, \ldots, x_{i-1})$.
2. If $s = f(x_1, \ldots, x_{i-1}, x'_i, \ldots, x'_n)$, then if $s = 1$ output y_i, and otherwise (i.e., if $s = 0$) go to Step 1 and repeat the sampling process.

The above attack is inspired by the two-party attack of [HO14]. Our main contribution is to do the following steps. (1) First, analyze this attack in the generalized tampering setting and show its power, which implies the multiparty case as special case. This already gives an alternative, and in our eyes simpler, proof of the classic result of [BOL89] (2) We then extend this attack and its analysis to the *real-output* setting. (3) Finally, we show how to approximate this attack in polynomial time.

2 Multi-Party Poisoning: Definitions and Main Results

Notation. We use bold font (e.g., $\mathbf{x}, \mathbf{S}, \boldsymbol{\alpha}$) to represent random variables, and usually use same non-bold letters for denoting samples from these distributions. We use $d \leftarrow \mathbf{d}$ to denote the process of sampling d from the random variable $\mathbf{d}$. By $\mathbb{E}[\boldsymbol{\alpha}]$ we mean the expected value of $\boldsymbol{\alpha}$ over the randomness of $\boldsymbol{\alpha}$, and by $\mathbb{V}[\boldsymbol{\alpha}]$ we denote the variance of random variable $\boldsymbol{\alpha}$. We might use a "processed" version of $\boldsymbol{\alpha}$, and use $\mathbb{E}[f(\boldsymbol{\alpha})]$ and $\mathbb{V}[f(\boldsymbol{\alpha})]$ to denote the expected value and variance, respectively, of $f(\boldsymbol{\alpha})$ over the randomness of $\boldsymbol{\alpha}$. A learning problem $(\mathcal{A}, \mathcal{B}, \mathbf{d}, \mathcal{H})$ is specified by the following components. The set $\mathcal{A}$ is the set of possible *instances*, $\mathcal{B}$ is the set of possible *labels*, $\mathbf{d}$ is distribution over $\mathcal{A} \times \mathcal{B}$.[1] The set $\mathcal{H} \subseteq \mathcal{B}^{\mathcal{A}}$ is called the *hypothesis space* or *hypothesis class*. An *example* s is a pair $s = (a, b)$ where $x \in \mathcal{A}$ and $y \in \mathcal{B}$. We consider *loss functions* $\mathsf{Loss}\colon \mathcal{B} \times \mathcal{B} \mapsto \mathbb{R}_+$ where $\mathsf{Loss}(b', b)$ measures how different the 'prediction' y' (of some possible hypothesis $h(a) = y'$) is from the true outcome y. We call a loss function *bounded* if it always takes values in $[0, 1]$. A natural loss function for classification tasks is to use $\mathsf{Loss}(b', b) = 0$ if $y = y'$ and $\mathsf{Loss}(b', b) = 1$ otherwise. The *risk* of a hypothesis $h \in \mathcal{C}$ is the expected loss of h with respect to $\mathbf{d}$, namely $\mathsf{Risk}(h) = \mathbb{E}_{(a,b)\leftarrow\mathbf{d}}[\mathsf{Loss}(h(a), b)]$. The *average error* which quantifies the total error of the protocol is defined as $\mathsf{Err}(\mathbf{d}) = \Pr_{h\leftarrow\Pi,(a,b)\leftarrow\mathbf{d}}[\mathsf{Loss}(h(a), b)]$.

[1] By using joint distributions over $\mathcal{A} \times \mathcal{B}$, we jointly model a set of distributions over $\mathcal{A}$ and a concept class mapping $\mathcal{A}$ to $\mathcal{B}$ (perhaps with noise and uncertainty).

Definition 2.1 (Multi-party learning protocols)**.** *An m-party learning protocol Π for the m-party learning problem $(\mathcal{D}, \mathcal{H})$ consists of an aggregator function G and m (interactive) data providers $\mathcal{P} = \{P_1, \ldots, P_m\}$. For each data provider P_i, there is a distribution $\mathbf{d}_i \in \mathcal{D}$ that models the (honest) distribution of labeled samples generated by P_i, and there is a final (test) distribution $\mathbf{d}$ that $\mathcal{P}, G$ want to learn jointly. The protocol runs in r rounds and at each round, based on the protocol Π, one particular data owner P_i broadcasts a single labeled example $(a, b) \leftarrow \mathbf{d}_i$.*[2] *In the last round, the aggregator function G maps the the messages to an output hypothesis $h \in \mathcal{H}$.*

Now, we define poisoning attackers that target multi-party protocols. We formalize a more general notion that includes p-tampering attacks and k-party corruption as special case.

Definition 2.2 (Multi-party (k, p)-poisoning attacks)**.** *A (k, p)-poisoning attack against an m-party learning protocol Π is defined by an adversary* Adv *who can control a subset $\mathcal{C} \subseteq [m]$ of the parties where $|\mathcal{C}| = k$. The attacker* Adv *shall pick the set $\mathcal{C}$ at the beginning. At each round j of the protocol, if a data provider $P_i \in \mathcal{C}$ is supposed to broadcast the next example from its distribution $\mathbf{d}_i$, the adversary can partially control this sample using the tampered distribution $\widetilde{\mathbf{d}}$ such that $|\widetilde{\mathbf{d}} - \mathbf{d}_i| \leq p$ in total variation distance. Note that the distribution $\widetilde{\mathbf{d}}$ can depend on the history of examples broadcast so far, but the requirement is that, conditioned on this history, the malicious message of adversary modeled by distribution $\widetilde{\mathbf{d}}$, is at most p-statistically far from $\mathbf{d}_i$. We use Π_{Adv} to denote the protocol in presence of* Adv*. We also define the following notions.* Adv *is a* plausible *adversary, if it always holds that* $\text{Supp}(\widetilde{\mathbf{d}}) \subseteq \text{Supp}(\mathbf{d}_i)$. Adv *is* efficient *if it runs in polynomial time in the total length of the messages exchanged during the protocol (from the beginning till end).*

Remark 2.3 (Static vs. adaptive corruption)**.** *Definition 2.2 focuses on corrupting k parties statically. A natural extension of this definition in which the set $\mathcal{C}$ is chosen* adaptively *[CFGN96] while the protocol is being executed can also be defined naturally. However, here we focus on static corruption and leave the possibility of improving our results in the adaptive case for future work.*

Remark 2.4 (Plausible vs. clean-label attacks)**.** *In recent years, the term clean-label is used for data-poisoning attacks that must use the correct label for the poison data. This definition is special case of plausibility as one can define the support set to be the all images with their correct label. However, our definition is more general and can be used for model poisoning attacks against federated learning as well. For example, the distribution $\mathbf{d}$ could be the distribution of gradients at a certain round. Being plausible in this setting means that the gradient must be calculated based on a real input and cannot be arbitrary.*

We now formally state our result about the power of (k, p)-poisoning attacks.

Theorem 2.5 (Power of efficient multi-party poisoning)**.** *In any m-party protocol Π for parties $\mathcal{P} = \{P_1, \ldots, P_m\}$, for any $p \in [0, 1]$ and $k \in [m]$, the following hold where M is the total length of the messages exchanged.*

1. *For any bad property $B : \mathcal{H} \to \{0, 1\}$, there is a* plausible *(k, p)-poisoning attack* Adv *that runs in time* $\text{poly}(M/\varepsilon)$ *and increases the probability of B from μ (in the no-attack setting) to*

$$\mu' \geq \mu^{1-p} - \varepsilon.$$

[2] We can directly model settings where more data is exchanged in one round, however, we stick to the simpler definition w.l.o.g.

2. *If the (normalized) loss function is bounded (i.e., it outputs in* $[0,1]$*), then there is a* plausible*,* (k,p)*-poisoning* Adv *that runs in time* $\mathrm{poly}(M/\varepsilon)$ *and increases the average error of the protocol as*

$$\begin{aligned}\mathsf{Err}_{\mathrm{Adv}}(\mathbf{d}) &\geq \mathsf{Err}(\mathbf{d})^{-p} \cdot \mathop{\mathbb{E}}_{h \leftarrow \Pi}[\mathsf{Risk}(h,\mathbf{d})^{1+p}] \\ &\geq \mathsf{Err}(\mathbf{d}) + \frac{p \cdot k}{2m} \cdot \nu - \varepsilon\end{aligned}$$

where $\nu = \mathbb{V}_{h \leftarrow \Pi}[\mathsf{Risk}(h,\mathbf{d})]$ *and* $\mathbb{V}[\cdot]$ *is the variance.*

Allowing different distributions in different rounds. In Definition 2.2, we restrict the adversary to remain "close" to $\mathbf{d}_i$ for each message sent out by one of the corrupted parties. A natural question is: what happens if we allow the parties distributions to be different in different rounds. For example, in a round j, a party P_i might send *multiple* training examples $D^{(j)} = \left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$, and we want to limit the *total* statistical distance between the distribution of the larger message $D^{(j)}$ from $\mathbf{d}_i^k$ (i.e., k iid samples from $\mathbf{d}_i$).[3] We emphasize that, our results extend to this more general setting as well. In particular, the proof of Theorem 2.5 directly extends to a more general setting where we can allow the honest distribution $\mathbf{d}_i$ of each party i to also depend on the round j in which these messages are sent. Thus, we can use a round-specific distribution $\mathbf{d}_i^{(j)}$ to model the joint distribution of *multiple* samples $D^{(j)} = \left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$ that are sent out in the j'th round by the party P_i. This way, we can obtain the stronger form of attacks that remain statistically close to the joint (correct) distribution of the (multi-sample) messages sent in a round. In fact, as we will discuss shortly $D^{(j)}$ might be of completely different type.

Allowing randomized aggregation. The aggregator G is a simple function that maps the transcript of the exchanged messages to a hypothesis h. A natural question is: what happens if we generalize this to the setting where G is allowed to be randomized. We note that in Theorem 2.5, Part 2 can allow G to be randomized, but Parts 1 and 3 need deterministic aggregation. The reason is that for those parts, we need the transcript to determine the confidence and average error functions. One general way to make up for randomized aggregation is to allow the parties to inject randomness into the transcript as they run the protocol by sending messages that are not necessarily learning samples from their distribution $\mathbf{d}_i$. As described above, our attacks extend to this more general setting as well. Otherwise, we will need the adversary to be able to also depend on the randomness of G, but that is also a reasonable assumption if the aggregation is used using public beacon that could be obtained by the adversary as well.

Before proving Theorem 2.5, we need to develop our main result about the power of generalized p-tampering attacks. In Section 3, we prove such result, and then in Section 3.2 we prove Theorem 2.5.

3 Multi-Party Poisoning via Generalized p-Tampering

To prove our Theorem 2.5 we interpret the multi-party learning protocol as a coin tossing protocol in which the final bit is 1 if h has the (bad) property B. We define a corresponding attack model

[3] Note that, even if each block in $\left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$ remains p-close to $\mathbf{d}_i$, their joint distribution could be quite far from $\mathbf{d}_i^k$.

in coin tossing protocols that can be directly used to obtain the desired goal; this model is called *generalized* p-tampering. Below, we formally state our main result about the power of generalized p-tampering attacks. We start by formalizing some notation and definitions.

Notation. By $\mathbf{x} \equiv \mathbf{y}$ we denote that the random variables $\mathbf{x}$ and $\mathbf{y}$ have the same distributions. Unless stated otherwise, by using a bar over a variable, we emphasize that it is a vector. By $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ we refer to a joint distribution over vectors with n components. For a joint distribution $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we use $\mathbf{x}_{\leq i}$ to denote the joint distribution of the first i variables $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_i)$. Also, for a vector $\overline{x} = (x_1 \ldots x_n)$ we use $x_{\leq i}$ to denote the prefix $(x_1, \ldots, x_i)$. For a randomized algorithm $L(\cdot)$, by $y \leftarrow L(x)$ we denote the randomized execution of L on input x outputting y. For a distribution $(\mathbf{x}, \mathbf{y})$, by $(\mathbf{x} \mid \mathbf{y})$ we denote the conditional distribution $(\mathbf{x} \mid \mathbf{y} = y)$. By $\mathrm{Supp}(\mathbf{d}) = \{d \mid \Pr[\mathbf{d} = d] > 0\}$ we denote the support set of $\mathbf{d}$. By $T^{\mathbf{d}}(\cdot)$ we denote an algorithm $T(\cdot)$ with oracle access to a sampler for $\mathbf{d}$ that upon every query returns fresh samples from $\mathbf{d}$. By $\mathbf{d}^n$ we denote the distribution that returns n iid samples from $\mathbf{d}$.

Definition 3.1 (Valid prefixes)**.** *Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an arbitrary joint distribution. We call $x_{\leq i} = (x_1, \ldots, x_i)$ a* valid prefix *for $\overline{\mathbf{x}}$ if there exist $x_{i+1}, \ldots, x_n$ such that $(x_1, \ldots, x_n) \in \mathrm{Supp}(\overline{\mathbf{x}})$. $\mathrm{ValPref}(\overline{\mathbf{x}})$ denotes the set of all valid prefixes of $\overline{\mathbf{x}}$.*

Definition 3.2 (Tampering with random processes)**.** *Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an arbitrary joint distribution. We call a (potentially randomized and possibly computationally unbounded) algorithm T an (online)* tampering algorithm for $\overline{\mathbf{x}}$ *if given any prefix $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, we have*

$$\Pr_{x_i \leftarrow \mathsf{T}(x_{\leq i-1})}[x_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{x}})] = 1 \ .$$

Namely, $\mathsf{T}(x_{\leq i-1})$ outputs x_i such that $x_{\leq i}$ is again a valid prefix. We call T an efficient *tampering algorithm for $\overline{\mathbf{x}}$ if it runs in time $\mathrm{poly}(N)$ where N is the bit length of $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.*

Definition 3.3 (Online samplers)**.** *We call OnSam an* online sampler *for $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ if for all $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, $\mathsf{OnSam}(n, x_{\leq i-1}) \equiv \mathbf{x}_i$. Moreover, we call $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$* online samplable *if it has an online sampler that runs in time $\mathrm{poly}(N)$ where N is the bit length of $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.*

Notation for tampering distributions. Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an arbitrary joint distribution and T a tampering algorithm for $\overline{\mathbf{x}}$. For any subset $S \subseteq [n]$, we define $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$ to be the joint distribution that is the result of online tampering of T over set S, where $\overline{\mathbf{y}} \equiv (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ is sampled inductively as follows. For every $i \in [n]$, suppose $y_{\leq i-1}$ is the previously sampled block. If $i \in S$, then the i^{th} block $\mathbf{y}_i$ is generated by the tampering algorithm $\mathsf{T}(y_{\leq i-1})$, and otherwise, $\mathbf{y}_i$ is sampled from $(\mathbf{x}_i \mid \mathbf{x}_{i-1} = y_{\leq i-1})$. For any *distribution* $\mathbf{S}$ over subsets of $[n]$, by $\langle \overline{\mathbf{x}} \parallel \mathsf{T}, \mathbf{S} \rangle$ we denote the random variable that can be sampled by first sampling $S \leftarrow \mathbf{S}$ and then $\overline{y} \leftarrow \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$.

Definition 3.4 (p-covering)**.** *Let $\mathbf{S}$ be a distribution over the subsets of $[n]$. We call $\mathbf{S}$ a* p-covering *distribution on $[n]$ (or simply p-covering, when n is clear from the context), if for all $i \in [n], \Pr_{S \leftarrow \mathbf{S}}[i \in S] = p$.*

The following theorem states the power of generalized p-tampering attacks.

Theorem 3.5 (Biasing of bounded functions through generalizing p-tampering)**.** *Let* $\mathbf{S}$ *be a* p*-covering distribution on* $[n]$*,* $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ *be a joint distribution,* $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$*, and* $\mu = \mathbb{E}[f(\overline{\mathbf{x}})]$*. Then, for any* $\varepsilon \in [0,1]$*, there exists a tampering algorithm* T_ε *that, given oracle access to* f *and any online sampler* OnSam *for* $\overline{\mathbf{x}}$*, it runs in time* $\mathrm{poly}(N/\varepsilon)$*, where* N *is the bit length of any* $\overline{x} \leftarrow \overline{\mathbf{x}}$*, and for* $\overline{\mathbf{y}}_\varepsilon \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f,\mathsf{OnSam}}, \mathbf{S} \rangle$*, it holds that*

$$\mathbb{E}\left[f(\overline{\mathbf{y}}_\varepsilon)\right] \geq \mu^{-p} \cdot \mathbb{E}\left[f(\overline{\mathbf{x}})^{1+p}\right] - \varepsilon \ .$$

Special case of Boolean functions. When the function f is Boolean, we get $\mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] = \mu^{1-p} \geq \mu(1 + \Omega_\mu(p))$, which matches the bound proved in [BOL89] for the special case of $p = k/n$ for integer $k \in [n]$ and for $\mathbf{S}$ that is uniformly random subset of $[n]$ of size k. (The same bound for the case of 2 parties was proved in [HO14] with extra properties). Even for this case, compared to [BOL89, HO14] our result is more general, as we can allow $\mathbf{S}$ with arbitrary $p \in [0,1]$ *and* achieve a polynomial time attack given oracle access to an online sampler for $\overline{\mathbf{x}}$. The work of [HO14] also deals with polynomial time attackers for the special case of 2 parties, but their efficient attackers use a different oracle (i.e., OWF inverter), and it is not clear whether or not their attack extend to the case of more then 2 parties. Finally, both [BOL89, HO14] prove their bound for the *geometric* mean of the averages for different $S \leftarrow \mathbf{S}$, while we do so for their arithmetic mean, but we emphasize that this is enough for all of our applications.

The bounds of Theorem 3.5 relies on the quantity $\mu' = \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}]$. A natural question is: how large is μ' compared to μ? As discussed above, for the case of Boolean f, we already know that $\mu' \geq \mu$, but that argument does not apply to the real-output f. A simple application of Jensen's inequality shows that $\mu \leq \mu'$ in general, but that still does not mean that $\mu' \gg \mu$.

General case of real-output functions: relating the bias to the variance. If $\mathbb{V}[f(\overline{\mathbf{x}})] = 0$, then no tampering attack can achieve any bias, so any the minimum bias of all attacks should somehow depend on the variance of $f(\overline{\mathbf{x}})$. In the following, we show that this gap does exist and that $\mu' - \mu \geq \Omega(p \cdot \mathbb{V}[f(\overline{\mathbf{x}})])$. Similar results relating the bias the the variance of the original distribution were previously proved [MDM18a, MM17, ACM+14] for the special case of p-tampering attacks (i.e., $\mathbf{S}$ chooses every $i \in [n]$ independently with probability p). Here, we obtain a more general result for any p-covering set structure $\mathbf{S}$.

Corollary 3.6. *If* $\nu = \mathbb{V}[f(\overline{\mathbf{x}})]$*, then the computationally bounded attack of Theorem 3.5 achieves*

$$\mathbb{E}\left[f(\overline{\mathbf{y}}_\varepsilon)\right] - \mathbb{E}\left[f(\overline{\mathbf{x}})\right] \geq \frac{p \cdot (p+1)}{2 \cdot \mu^p} \cdot \nu - \varepsilon \geq \frac{p}{2} \cdot \nu - \varepsilon \ .$$

We prove this corollary in Appendix A.1 by proving a connection between the bound of Theorem 3.5 and the variance ν.

3.1 Proving Theorem 3.5

Here, we first prove the power of computationally unbounded adversaries. Then, we show how we can "approximate" this attack with a polynomial-time adversary and get almost the same bias.

3.1.1 Warm up: Computationally Unbounded Adversaries

The construction below describes a computationally unbounded biasing algorithm that achieves the bounds of Theorem 3.5.

Construction 3.7 (Rejection-sampling tampering)**.** *Let* $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ *and* $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$*. The* rejection sampling *tampering algorithm* RejSam^f *works as follows. Given the valid prefix* $y_{\le i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$*, the tampering algorithm would do the following:*

1. *Sample* $y_{\ge i} \leftarrow (\mathbf{x}_{\ge i} \mid y_{\le i-1})$ *by using the online sampler for* f*.*
2. *If* $s = f(y_1, \ldots, y_n)$*, then with probability* s *output* y_i*, otherwise go to Step 1 and repeat.*

We will first prove a property of the rejection sampling algorithm when applied on every block.

Definition 3.8 (Notation for partial expectations of functions)**.** *Suppose* $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto \mathbb{R}$ *is defined over a joint distribution* $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$*,* $i \in [n]$*, and* $x_{\le i} \in \mathrm{ValPref}(\overline{\mathbf{x}})$*. Then, using a small hat, we define the notation* $\hat{f}(x_{\le i}) = \mathbb{E}_{\overline{x} \leftarrow (\overline{\mathbf{x}} | x_{\le i})}[f(\overline{x})]$*. E.g., for* $\overline{x} = x_{[n]}$*, we have* $\hat{f}(\overline{x}) = f(\overline{x})$*.*

Claim 3.9. *If* $\langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}^f, [n] \rangle \equiv \overline{\mathbf{y}}^{[n]} \equiv (\mathbf{y}_1, \ldots, \mathbf{y}_n)$*. Then, for every valid* $y_{\le i} \in \mathrm{ValPref}[\overline{\mathbf{x}}]$*,*

$$\frac{\Pr[\mathbf{y}_{\le i} = y_{\le i}]}{\Pr[\mathbf{x}_{\le i} = y_{\le i}]} = \frac{\hat{f}(y_{\le i})}{\mu} \ .$$

Proof. Based on the description of RejSam^f, for any $y_{\le i} \in \mathrm{ValPref}(\overline{\mathbf{x}})$ the following equation holds for the probability of sampling y_i conditioned on prefix $y_{\le i-1}$.

$$\begin{aligned}\Pr[\mathbf{y}_i = y_i \mid y_{\le i-1}] = \Pr[\mathbf{x}_i = y_i \mid y_{\le i-1}] \cdot \hat{f}(y_{\le i}) \\ + (1 - \hat{f}(y_{\le i-1})) \cdot \Pr[\mathbf{y}_i = y_i \mid y_{\le i-1}].\end{aligned}$$

The first term in this equation corresponds to the probability of selecting and accepting in the first round of sampling and the second term corresponds to the probability of selecting and accepting in any round except the first round. Therefore we have

$$\Pr[\mathbf{y}_i = y_i \mid y_{\le i-1}] = \frac{\hat{f}(y_{\le i})}{\hat{f}(y_{\le i-1})} \cdot \Pr[\mathbf{x}_i = y_i \mid y_{\le i-1}] \ ,$$

which implies that

$$\begin{aligned}\Pr[\mathbf{y}_{\le i} = y_{\le i}] &= \prod_{j \in [i]} \left(\frac{\hat{f}(y_{\le j})}{\hat{f}(y_{\le j-1})} \right) \cdot \Pr[\mathbf{x}_{\le i} = y_{\le i}] \\ &= \frac{\hat{f}(y_{\le i})}{\mu} \cdot \Pr[\mathbf{x}_{\le i} = y_{\le i}] \ .\end{aligned}$$

□

Now, we prove two properties for *any* tampering algorithm (not just rejection sampling) over a p-covering distribution.

Lemma 3.10. *Let* $\mathbf{S}$ *be p-covering for* $[n]$ *and* $\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})$*. For any* $S \in \mathrm{Supp}(\mathbf{S})$ *and an arbitrary tampering algorithm* T *for* $\overline{\mathbf{x}}$*, let* $\overline{\mathbf{y}}^S \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$*. Then,*

$$\prod_{S \in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{\Pr[\mathbf{S}=S]} = \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p .$$

Proof. For every $y_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{y}}^{[n]}) \subseteq \mathrm{ValPref}(\overline{\mathbf{x}})$ define $\rho[y_{\leq i}]$ as

$$\rho[y_{\leq i}] = \frac{\Pr[\mathbf{y}_i^{[n]} = x_i \mid \mathbf{y}_{\leq i-1}^{[n]} = y_{\leq i-1}]}{\Pr[\mathbf{x}_i = x_i \mid \mathbf{x}_{\leq i-1} = y_{\leq i-1}]} .$$

Then, for all $\overline{y} \in \mathrm{ValPref}(\overline{\mathbf{y}}^S) \subseteq \mathrm{ValPref}(\overline{\mathbf{x}})$ we have

$$\Pr[\overline{\mathbf{y}}^S = y] = \Pr[\overline{\mathbf{x}} = y] \cdot \prod_{i \in S} \rho[y_{\leq i}] .$$

Therefore we have

$$\prod_{S \in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{\Pr[\mathbf{S}=S]} = \left(\prod_{i \in [n]} \rho[y_{\leq i}] \right)^p .$$

□

Claim 3.11. *Suppose* $\mathbf{S}$ *is p-covering on* $[n]$*,* $\overline{\mathbf{y}}^S \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$ *for any* $S \leftarrow \mathbf{S}$*, and* $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, \mathbf{S} \rangle$ *for an arbitrary tampering algorithm* T *for* $\overline{\mathbf{x}}$*. Then, it holds that*

$$\mathbb{E}[f(\overline{\mathbf{y}})] \geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p .$$

Proof. Let $h_{S,\overline{y}} = \frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}$. Also let $\mathcal{Z} \subseteq \mathrm{Supp}(\overline{\mathbf{x}})$. Note that $\mathrm{Supp}(\overline{\mathbf{y}}^S) \subseteq \mathcal{Z}$ for any $S \subseteq [n]$. Therefore, we have $\mathbb{E}[f(\overline{\mathbf{y}})] = \mathbb{E}_{S \leftarrow \mathbf{S}} \, \mathbb{E}_{\overline{y} \leftarrow \overline{\mathbf{y}}^S} [f(y)]$ is equal to

$$\begin{aligned}
&\sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{y}}^S = \overline{y}] \cdot f(\overline{y}) \\
&= \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \sum_{\overline{y} \in \mathcal{Z}} h_{S,\overline{y}} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot h_{S,\overline{y}} \\
&\qquad \text{(by AM-GM inequality)} \\
&\geq \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \prod_{S \in 2^{[n]}} h_{S,\overline{y}}^{\Pr[\mathbf{S}=S]} \\
&\qquad \text{(by } p\text{-covering of } \mathbf{S} \text{ and Lemma 3.10)} \\
&= \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p .
\end{aligned}$$

□

We now prove the main result using the one-rejection sampling tampering algorithm and also relying on the p-covering property of $\mathbf{S}$. In particular, if $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}^f, \mathbf{S} \rangle$, then by Claims 3.11 and 3.9 we have

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{y}})] &\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&\quad \text{(by Claim 3.9)} \\
&= \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu} \right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= \mu^{-p} \cdot \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y})^{1+p} \\
&= \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] \ .
\end{aligned}$$

3.1.2 Proving Theorem 3.5 for Polynomially Bounded Attacks

In this section, we prove the second item of Theorem 3.5. Namely, we show an efficient tampering algorithm whose average is ε-close to the average of RejSam. We define this attack as follows:

Construction 3.12 (k-rejection-sampling tampering)**.** *Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f \colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. The k-*rejection sampling *tampering algorithm RejSam_k^f works as follows. Given the valid prefix $y_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, the tampering algorithm would do the following for k times:*

1. *Sample $y_{\geq i} \leftarrow (\mathbf{x}_{\geq i} \mid y_{\leq i-1})$ by using the online sampler for f.*
2. *Let $s = f(y_1, \ldots, y_n)$; with probability s output y_i, otherwise go to Step 1.*

If no y_i was output during any of the above k iterations then output a fresh sample $y_i \leftarrow (\mathbf{x}_i \mid y_{\leq i-1})$.

The output distribution of RejSam_k on any input, converges to the rejections sampling tampering algorithm RejSam for sufficiently large $k \to \infty$.

Notation. Below, use the notation $\overline{\mathbf{z}} = \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}_k^f, S \rangle$ and $\mu_k = \mathbb{E}[f(\overline{\mathbf{z}})]$.

We will prove the following claim which will directly completes the proof of second part of Theorem 3.5.

Claim 3.13. *Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f \colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. For any $\varepsilon \in [0,1]$, let $k \geq \frac{16 \ln(2n/\varepsilon)}{\varepsilon^2 \mu^2}$. Then RejSam_k runs in time $O(k) = \mathrm{poly}(N/(\varepsilon \cdot \mu))$, where $N \geq n$ is the total bit-length of representing $\overline{\mathbf{x}}$, and for $\overline{\mathbf{z}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}_k^{f,\mathsf{OnSam}}, \mathbf{S} \rangle$ it holds that*

$$\mathbb{E}[f(\overline{\mathbf{z}})] \geq \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] - \varepsilon \ .$$

Proof. It is easy to see why RejSam_k runs in time $O(k)$ and thus we will focus on proving the expected value of the output of the k-rejection sampling tampering algorithm. To that end, we start by providing some definitions relevant to our analysis.

Definition 3.14. *For $\delta \geq 0$, let*

$$\mathrm{High}(\delta) = \{\overline{x} \mid \overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}}) \wedge \forall i \in [n], \hat{f}(x_{\leq i-1}) \geq \delta\},\ \mathrm{Low}(\delta) = \mathrm{Supp}(\overline{\mathbf{x}}) \setminus \mathrm{High}(\delta)\ ,$$

$$\mathrm{Big}(\delta) = \{\overline{x} \mid \overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}}) \wedge f(\overline{x}) \geq \delta\},\ \textit{and}\ \ \mathrm{Small}(\delta) = \mathrm{Supp}(\overline{\mathbf{x}}) \setminus \mathrm{Big}(\delta)\ .$$

Claim 3.15. *For $\delta_1 \cdot \delta_2 = \delta$, it holds that*

$$\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2\ .$$

As a result, it holds that $\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \wedge \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2$, and so

$$\sum_{\overline{x} \in \mathrm{Big}(\delta_1) \cap \mathrm{Low}(\delta)} \Pr[\overline{x} = \overline{\mathbf{x}}] \leq \delta_2\ .$$

Proof. Let $t\colon \mathrm{Low}(\delta) \to \mathrm{ValPref}(\overline{\mathbf{x}})$ be such that $t(\overline{x})$ is the smallest prefix $x_{\leq i}$ such that $\hat{f}(x_{\leq i}) \leq \delta$. Now consider the set $T = \{t(\overline{x}) \mid \overline{x} \in \mathrm{Low}(\delta)\}$. For any $w \in T$ we have

$$\delta \geq \hat{f}(w) \geq \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid t(\overline{x}) = w] \cdot \delta_1\ ,$$

which implies

$$\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid t(\overline{x}) = w] \leq \delta_2\ .$$

Thus, we have

$$\begin{aligned}
&\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&= \sum_{w \in T} \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \wedge t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&= \sum_{w \in T} \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta) \wedge t(\overline{x}) = w] \cdot \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&\leq \sum_{w \in T} \delta_2 \cdot \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2\ .
\end{aligned}$$

□

Claim 3.16. *Let $x \in \mathrm{High}(\delta)$, then we have*

$$\Pr[\overline{\mathbf{z}} = \overline{y}] \geq (1 - (1-\delta)^k)^n \cdot \frac{f(\overline{y})}{\mu} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}]\ .$$

Proof. Consider $\mathrm{E}_{k,y_{\leq i}}$ to be the event that RejSam_k outputs one of its first k samples, when performed on $y_{\leq i}$. Then, it holds that

$$\Pr[\mathrm{E}_{k,y_{\leq i}}] = 1 - (1 - \hat{f}(y_{\leq i}))^k \geq 1 - (1-\delta)^k\ .$$

On the other hand, we know that $\Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\leq i} \wedge \mathrm{E}_{k,y_{\leq i}}] = \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\leq i}]$. Thus, we have

$$\begin{aligned}
\Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\leq i}] &\geq \Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\leq i} \wedge \mathrm{E}_{k,y_{\leq i}}] \cdot \Pr[\mathrm{E}_{k,y_{\leq i}}] \\
&= \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\leq i}] \cdot \Pr[\mathrm{E}_{k,y_{\leq i}}] \\
&\geq \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\leq i}] \cdot (1 - (1-\delta)^k)^n.
\end{aligned}$$

By multiplying these inequalities for $i \in [n]$ we get $\Pr[\overline{\mathbf{z}} = \overline{y}] \geq (1 - (1-\delta)^k)^n \cdot \Pr[\overline{\mathbf{y}} = \overline{x}]$. □

Claim 3.17. *For $\delta_1 \cdot \delta_2 = \delta$, it holds that*

$$\mu_k \geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\delta_1 + \delta_2}{\mu} - n \cdot (1-\delta)^k \ .$$

Proof. Let

$$\mu' = \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Small}(\delta_1)} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \ ,$$

$$\text{and} \quad \mu'' = \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Big}(\delta_1)} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \ .$$

By Claim, 3.11 we have

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{z}})] &\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{\Pr[\overline{\mathbf{z}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&\geq \sum_{\overline{y} \in \mathrm{High}(\delta)} \left(\frac{\Pr[\overline{\mathbf{z}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
\text{(by Claim 3.16)} \quad &\geq \sum_{\overline{y} \in \mathrm{High}(\delta)} (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \mu' - \mu'' \right) .
\end{aligned}$$

We have $\mu' \leq \delta_1^{1+p}/\mu^p \leq \delta_1/\mu$, because $f(\overline{y}) \leq \delta_1$ for all $\overline{y} \in \mathrm{Small}(\delta_1)$. Also, by Claim 3.15, we get

$$\mu'' \leq \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Big}(\delta_1)} \left(\frac{1}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \leq \frac{\delta_2}{\mu^p} \leq \frac{\delta_2}{\mu} \ .$$

Therefore, we have

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{z}})] &\geq (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\delta_1 + \delta_2}{\mu} \right) \\
\text{(by Bernoulli inequality)} \quad &\geq (1 - n \cdot (1-\delta)^k) \cdot \left(\sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\delta_1 + \delta_2}{\mu} \right) \\
&\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\delta_1 + \delta_2}{\mu} - n \cdot (1-\delta)^k .
\end{aligned}$$

□

In order to conclude the proof of Claim 3.13, we can set $\delta_1 = \delta_2 = \sqrt{\delta}$ and let $\delta \leq (\varepsilon\mu/4)^2$. Then, given that we have $k \geq \frac{16\ln(2n/\varepsilon)}{\varepsilon^2\mu^2}$, we get

$$\mathbb{E}[f(\overline{\mathbf{z}})] \geq \sum_{\overline{y}\in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\varepsilon}{2} - \frac{\varepsilon}{2} \,.$$

□

3.2 Obtaining (k, p)-Poisoning: Proof of Theorem 2.5 using Theorem 3.5

In this section, we formally prove Theorem 2.5 using Theorems 3.5. We first prove the first part of theorem about the boolean property.

Proof of Theorem 2.5, Part 1. For a subset $C \subseteq [m]$ let $P_C = \{P_i; i \in C\}$ and R_C be the subset of rounds where one of the parties in P_C sends an example. Also for a subset $S \subseteq [n]$, we define $\mathbf{Bion}(S, p)$ to be a distribution over all the subsets of S, where each subset $S' \subseteq S$ hast the probability $p^{|S'|} \cdot (1-p)^{|S|-|S'|}$. Now, consider the covering $\mathbf{S}$ of the set $[n]$ which is distributed equivalent to the following process. First sample a uniform subset C of $[m]$ of size k. Then sample and output a set S sampled from $\mathbf{Bion}(R_C, p)$. $\mathbf{S}$ is clearly a $(p \cdot \frac{k}{m})$-covering. We use this covering to prove the theorem. For $j \in [n]$ let $w(j)$ be the index of the provider at round j and let $\mathbf{d}_{w(j)}$ be the designated distribution of the jth round and let $\overline{\mathbf{x}} = \mathbf{d}_{w(1)} \times \cdots \times \mathbf{d}_{w(n)}$.

We define a function $f : \mathrm{Supp}(\overline{\mathbf{x}}) \to \{0, 1\}$, which is a Boolean function and is 1 if the output of the protocol has the property B, and otherwise it is 0. Now we use Theorem 3.5. We know that $\mathbf{S}$ is a $(p \cdot \frac{k}{m})$-covering for $[n]$. Therefore of Theorem 3.5, there exist an $\mathrm{poly}(m/\varepsilon)$ time tampering algorithm T_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f,\mathsf{OnSam}}, \mathbf{S}\rangle$ where $\mathbb{E}[f(\overline{\mathbf{y}})] \geq \mathbb{E}[f(\overline{\mathbf{y}})]^{1-pk/m} - \varepsilon$.

By an averaging argument, we can conclude that there exist a set $C \in [m]$ of size k for which the distribution $\mathbf{Bion}(R_C, p)$ produces average output at least $\mathbb{E}[f(\overline{\mathbf{y}})]^{1-pk/m} - \varepsilon$. Note that the measure of empty set in $\mathbf{Bion}(R_C, p)$ is exactly equal to $1 - p$ which means with probability $1 - p$ the adversary will not tamper with any of the blocks, therefore, the statistical distance $|\overline{\mathbf{x}} - \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f,\mathsf{OnSam}}, \mathbf{Bion}(R_C, p)\rangle|$ is at most p. This concludes the proof. □

Now we prove the second part using Theorem 3.5 and Lemma A.3.

Proof of Theorem 2.5 part 2. Now we prove the second part. The second part is very similar to first part except that the function that we define here is a real valued function. Consider the function $f_2 : \mathrm{Supp}(\overline{\mathbf{x}}) \to [0, 1]$ which is defined to be the risk of the output hypotheses. Now by Theorem 3.5 and Lemma A.3, we know that there is tampering algorithm T_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f_2,\mathsf{OnSam}}, \mathbf{S}\rangle$ such that

$$\mathbb{E}[f_2(\overline{\mathbf{y}})] \geq \mu_2 + \frac{p \cdot k}{2m} \cdot \nu - \varepsilon.$$

By a similar averaging argument we can conclude the proof. □

References

[ACM+14] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In *International Cryptology Conference*, pages 462–479. Springer, 2014.

[ACM+17] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. *Algorithmica*, 79(4):1052–1101, Dec 2017.

[BBV08] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 671–680. ACM, 2008.

[BCMC18] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. *arXiv preprint arXiv:1811.12470*, 2018.

[BDLS17] Sivaraman Balakrishnan, Simon S Du, Jerry Li, and Aarti Singh. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*, pages 169–212, 2017.

[BEK02] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002.

[BGS+17] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[BIK+17] 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.

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1467–1474. Omnipress, 2012.

[BNS+06] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

[BOL89] M. Ben-Or and N. Linial. Collective coin flipping. *Randomness and Computation*, 5:91–115, 1989.

[BVH+18] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648. ACM, 1996.

[CSV17] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60. ACM, 2017.

[CSX17] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

[CV19] Gregory Cirincione and Dinesh Verma. Federated machine learning for multi-domain operations at the tactical edge. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100606. International Society for Optics and Photonics, 2019.

[CWCP18] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pages 902–911, 2018.

[DKK+16] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 655–664. IEEE, 2016.

[DKK+18] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018.

[DKS17] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional Gaussians and Gaussian mixtures. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 73–84. IEEE, 2017.

[DKS18a] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical Gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1060. ACM, 2018.

[DKS18b] Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. *arXiv preprint arXiv:1806.00040*, 2018.

[FYB18] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[GR+18] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3518–3527, 2018.

[HO14] Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014.

[HO18] Jamie Hayes and Olga Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. In *Advances in Neural Information Processing Systems*, pages 6604–6615, 2018.

[HZ19] Yufei Han and Xiangliang Zhang. Robust federated training via collaborative machine teaching using trusted instances. *arXiv preprint arXiv:1905.02941*, 2019.

[KL93] Michael J. Kearns and Ming Li. Learning in the Presence of Malicious Errors. *SIAM J. on Computing*, 22(4):807–837, 1993.

[KMY+16] 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.

[KSL18] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

[LB17] JG Liao and Arthur Berg. Sharpening Jensen's inequality. *The American Statistician*, 2017.

[LRV16] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 665–674. IEEE, 2016.

[MDM18a] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-tampering attacks. In *Algorithmic Learning Theory*, pages 572–596, 2018.

[MDM18b] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-Tampering Attacks. In *ALT*, pages 572–596, 2018.

[MM17] Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-Tampering Attacks on Cryptographic Primitives, Extractors, and Learners. In *Theory of Cryptography Conference*, pages 245–279. Springer, 2017.

[MMR+16] 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.

[MR17] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 2017.

[PMSW16] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[PSBR18] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

[SKL17] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[STS16] Shiqi Shen, Shruti Tople, and Prateek Saxena. A uror: defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519. ACM, 2016.

[TCC19] Richard Tomsett, Kevin Chan, and Supriyo Chakraborty. Model poisoning attacks against distributed machine learning systems. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 110061D. International Society for Optics and Photonics, 2019.

[Val85] Leslie G. Valiant. Learning disjunctions of conjunctions. In *IJCAI*, pages 560–566, 1985.

[YCRB18] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

A Some Useful Inequalities

The following well-known variant of the inequality for the arithmetic mean and the geometric mean could be derived from the Jensen's inequality.

Lemma A.1 (Weighted AM-GM inequality)**.** *For any $n \in \mathbb{N}$, let $z_1, ..., z_n$ be a sequence of non-negative real numbers and let $w_1, ..., w_n$ be such that $w_i \geq 0$ for every $i \in [n]$ and $\sum_{i=1}^{n} w_i = 1$. Then, it holds that*

$$\sum_{i=1}^{n} w_i z_i \geq \prod_{i=1}^{n} z_i^{w_i} \ .$$

The following lemma provides a tool for lower bounding the gap between the two sides of Jensen's inequality, also known as the Jensen gap.

Lemma A.2 (Lower bound for Jensen gap [LB17])**.** *Let $\boldsymbol{\alpha}$ be a real-valued random variable,* $\mathrm{Supp}(\boldsymbol{\alpha}) \subseteq [0,1]$*, and $\mathbb{E}[\boldsymbol{\alpha}] = \mu$. Let $\varphi(\cdot)$ be twice differentiable on $[0,1]$, and let $h_b(a) = \frac{\varphi(a)-\varphi(b)}{(a-b)^2} - \frac{\varphi'(a)}{a-b}$. Then,*

$$\mathbb{E}[\varphi(\boldsymbol{\alpha})] - \varphi(\mu) \geq \mathbb{V}[\boldsymbol{\alpha}] \cdot \inf_{a \in [0,1]} \{h_\mu(a)\} \ .$$

A.1 Relating the Bias to the Variance

We first prove a lemma that shows the connection of bias to variance. Then, using this lemma we immediately get a $\Omega(p \cdot \mathbb{V}[f(\overline{\mathbf{x}})])$ lower bounds for the bias achieved by the attacker of Theorem 3.5 for the general case of real-valued functions and arbitrary p-covering set distribution $\mathbf{S}$.

Lemma A.3. *Let $\boldsymbol{\alpha}$ be any real-valued random variable over* $\mathrm{Supp}(\boldsymbol{\alpha}) \subseteq [0,1]$*, and* $p \in [0,1]$*. Let* $\mu = \mathbb{E}[\boldsymbol{\alpha}]$ *be the expected value of* $\boldsymbol{\alpha}$*,* $\nu = \mathbb{V}[\boldsymbol{\alpha}]$ *be the variance of* $\boldsymbol{\alpha}$*. Then, it holds that*

$$\mu^{-p} \cdot \mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu \geq \frac{p \cdot (p+1)}{2 \cdot \mu^p} \cdot \nu \geq \frac{p}{2} \cdot \nu \ .$$

Proof. We use Lemma A.2 by letting $\varphi(x) = x^{1+p}$. Thus, we have to minimize the following function on $x \in [0,1]$,

$$g_\mu(x) = \left(x^{1+p} - \mu^{1+p} - (1+p) \cdot \mu^p \cdot (x - \mu)\right) / (x-\mu)^2 \ .$$

We now prove that the minimum happens on $x = 1$. Note that the function $g_\mu(x)$ is continues on $[0, \mu)$ and $(\mu, 0]$ and the limit exists at $x = \mu$ and is equal to $1/2 \cdot p \cdot (1+p) \cdot \mu^{-1+p}$. Therefore if we show that g'_μ is negative for $x \in [0,\mu) \cup (\mu, 1]$ it implies that, $\forall x \in [0,1] g(x) \geq g(1)$. We have

$$\begin{aligned} g'_\mu(x) &= \frac{(p-1) \cdot \mu^{p+1} - (p+1) \cdot x \cdot \mu^p + (p+1) \cdot \mu \cdot x^p - (p-1) \cdot x^{p+1}}{(\mu - x)^3} \\ \text{(using } c = x/\mu) \quad &= \mu^{p-2} \cdot \frac{(p-1) - (p+1) \cdot c + (p+1) \cdot c^p - (p-1) \cdot c^{p+1}}{(1-c)^3} \ . \end{aligned}$$

We prove that the numerator $q(c) = (p-1) - (p+1) \cdot c + (p+1) \cdot c^p - (p-1) \cdot c^{p+1}$ is positive for $c > 1$ and negative for $0 < c < 1$. For $c > 0$, we have

$$\begin{aligned} q'(c) &= -(1+p) + (p+1) \cdot p \cdot c^{p-1} + (1-p) \cdot (p+1) \cdot c^p \\ &= (1+p) \cdot (p \cdot c^{p-1} + (1-p) \cdot c^p - 1) \\ \text{(by AM-GM inequality of Lemma A.1)} &\geq (1+p) \cdot (c^{p \cdot (p-1)} \cdot c^{(1-p) \cdot p} - 1) \\ &= 0 \ . \end{aligned}$$

Therefore, q is increasing for $c > 0$ which implies $\forall c \in [0,1], q(c) < q(1) = 0$ and $\forall c > 1, q(c) > q(1) = 0$. We have $\forall x \in [0,\mu) \cup (\mu, 1], g'(x) \leq 0$. Therefore we have

$$\forall x \in [0,1], g_u(x) \geq g_u(1) \ . \tag{1}$$

Now we prove that $g_\mu(1) \geq \frac{p(1+p)}{2}$. Consider the following function,

$$w(\mu) = g_\mu(1) = \left(1 - \mu^{1+p} - (1+p) \cdot \mu^p \cdot (1-\mu)\right) / (1-\mu)^2 \ .$$

We will show that q is a decreasing function for $\mu \in [0,1]$. We have

$$w'(\mu) = \frac{p \cdot (1-\mu^2) \cdot \mu^{p-1} + p^2 (1-\mu)^2 \cdot \mu^{p-1} + 2 \cdot (\mu^p - 1))}{(-1+\mu)^3} \ .$$

We will show that the numerator $s(\mu) = p \cdot (1-\mu^2) \cdot \mu^{p-1} + p^2 (1-\mu)^2 \cdot \mu^{p-1} + 2 \cdot (\mu^p - 1)$ is negative for $\mu \in [0,1]$. We have $s'(\mu) = p(p^2-1) \cdot (1-\mu)^2 \cdot \mu^{p-2}$ which is negative for $\mu \in [0,1]$. This implies that $\forall \mu \in [0,1], s(\mu) \geq s(1) = 0$. Therefore, w is a decreasing function, and we obtain

$$\forall \mu \in [0,1], g_\mu(1) = w(\mu) \geq \lim_{u \to 1} w(u) = \frac{p(1+p)}{2} \ . \tag{2}$$

Now, we conclude that

$$\begin{aligned}
\mu^{-p} \cdot \mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu &= \mu^{-p} \left(\mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu^{1+p} \right) \\
\text{(by Lemma A.2)} \quad &\geq \mu^{-p} \left(\inf_{x \in [0,1]} \{ g_\mu(x) \} \cdot \nu \right) \\
\text{(by Inequality 1)} \quad &\geq \mu^{-p} \cdot g_\mu(1) \cdot \nu \\
\text{(by Inequality 2)} \quad &\geq \frac{p \cdot (1+p)}{2 \cdot \mu^p} \cdot \nu \ .
\end{aligned}$$

□

Now, using Lemma A.3 and 3.5, we immidiately get Corollary 3.6